# DESIGN OF AN AUTOMOBILE ACCELERATOR/BRAKE PEDAL ROBOT FOR ADVANCED DRIVER ASSISTANCE SYSTEMS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Jake S. Schwartz

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

August 2017

Purdue University

Indianapolis, Indiana

# THE PURDUE UNIVERSITY GRADUATE SCHOOL
# STATEMENT OF COMMITTEE APPROVAL

Dr. Yaobin Chen, Chair

Department of Electrical and Computer Engineering

Dr. Yi Qiang

Department of Electrical and Computer Engineering

Dr. Lingxi Li

Department of Electrical and Computer Engineering

Dr. Stanley Yung-Ping Chien

Department of Electrical and Computer Engineering

**Approved by:**

Dr. Brian S. King

Head of the Graduate Program

I dedicate this to my parents and my siblings; Danke für alles.

# ACKNOWLEDGMENTS

I would like to thank Dr. Stanley Chien for introducing me to my thesis topic and for helping with my OBDII speed problem. Thanks Dr. Lingxi Li for your encouragement and positive attitude. It is contagious and appreciated. Thanks to Dr. Qiang Yi for theoretical and technical advise and expertises. It was helpful in modeling my thesis. Lastly, thanks to my advisor, Dr. Yaobin Chen, for all the emails, meetings, advice, and suggestions. Thanks for the guidance over the past 4 semesters!

I would like to thank Sherrie Tucker for her help throughout my graduate career; from the graduate school application process to graduation. Thanks for all the emails, information, reminders, and helping my stay on top of deadlines. Lastly, thanks for helping me with my last minute POS. It was much appreciated!

Finally, I would like to thank my family for their continued love and support throughout my life. Thanks to my parents for their unwavering faith in all my endeavors. A special thanks to my sister, Katie, for her help in the last year of my graduate studies. Thanks for all the food, for all the errands and tasks you have done for me so I could complete my thesis on time. And finally, thanks for the moral support throughout this past year.

TABLE OF CONTENTS

Page

# LIST OF TABLES

# LIST OF FIGURES

Figure Page

# ABBREVIATIONS

| | |
|---|---|
| PAEB | Pedestrian Automatic Emergency Braking |
| ADAS | Advanced Driver Assistance Systems |
| Car 1 | 2007 Ford Focus |
| Car 2 | 2008 Toyota Prius |

# ABSTRACT

Schwartz, Jake S. M.S.E.C.E., Purdue University, August 2017. Design of an Automobile Accelerator/Brake Pedal Robot for Advanced Driver Assistance Systems. Major Professor: Yaobin Chen.

This paper delves into designing an actuator system to control the accelerator and brake pedal to control the speed of an automobile. The actuator system comprises of an electric actuator that controls the accelerator pedal and a parallel linkage that controls the brake pedal. The parallel linkage is connected to the actuator such that it provides an opposite reaction to the brake pedal. This paper compares the speed control with and without the use of a parallel linkage with respect to overshoot and steady state error. A simplified actuator and car model are developed. A PID controller and a Fuzzy controller were designed, simulated, and compared in their ability to control the developed car model. Both controllers were then implemented and tested in two different cars.

# 1. INTRODUCTION

## 1.1 Background

As vehicles become more and more advanced, safety becomes a bigger concern. Features like Lane-Keep (lane-assist), Blind-Spot Detection, Adaptive Cruise Control, and Pedestrian Automatic Emergency Braking features all fall under a broad category called Advanced Driver Assistance Systems (ADAS). When new ADAS functionally is added or changed the car must go through long and rigorous testing to ensure that the features are safe for consumer use. One feature that is focused on is the Pedestrian Automatic Emergency Braking (PAEB) system. This is a type of collision avoidance and collision mitigation system. Its goal is to avoid a collision by reducing vehicle speed. If a collision is unavoidable, the goal becomes to mitigate or reduce the severity of the collision by reducing vehicle speed. The performance testing evaluates the PAEB system's ability to avoid collisions at a range of speeds. One obstacle in the way of this testing and analysis is accurate vehicle speed. It is difficult to analyze the performance if the vehicle speed is not at the desired speed during the testing. It can be more frustrating if the actual vehicle speed is not consistent for a given set point. This problem gives rise to the need for an accurate vehicle speed controller.

The cruise control system is a very accurate vehicle controller and it has been perfected over the past couple of decades. It is robust in that it accounts for flat terrain as well as for rolling hills. However, there are two issues with using cruise control for this testing. One is that when cruise control is active it often deactivates the PAEB system and secondly, vehicles on the market normally do not go below 20 mph.

The first issue is self explanatory. Both the Volvo S60 and Lexus LX460 have the PAEB functionality, and in both cars it is disabled when the cruise control system is

active. However, even if the cruise control could be used in the testing of the PAEB system, there is another issue: the cruise control function is disabled in most cars for speeds below 20 mph. For example, in many GM cars the lower limit is 25 mph [1] and in many Ford cars it is 20 mph [2]. It's not clear why automakers do this, but one argument is that it is a safety hazard in school zones. The logic behind this idea states that if the driver has set the cruise control, his/her foot is most likely not on the accelerator and is relaxed on the floorboard. If a child were to run in the path of the driver, the reaction time of hitting the brakes would be slower. On the other hand, if the driver is actively controlling the speed with his/her foot, the reaction time of releasing the accelerator and applying the brakes is faster, thus reducing the probably of a collision. Another argument is that it is dangerous to set the cruise control speed on highways and interstates where the speed limit is much higher. This large speed difference could increase the probability of an accident.

While this is a good safety feature it produces a hindrance when it comes to the testing of PAEB systems. As one might guess, using the human foot for vehicle speed control leads to inconsistencies from test to test as well as inaccurate speed control.

This brings about the need for a vehicle speed control system that is precise and accurate. However this system also needs to be dynamic enough to adapt to many different vehicle makes and models with different vehicle characteristics and dynamics. This desired capability arises from the fact that PAEB testing needs to be done in a wide range of vehicles and the controller is moved and installed from vehicle to vehicle. Having a system that automatically adapts to the vehicle characteristics would decrease setup time for the testing.

AB Dynamics has built a similar system. They have a product called Combined Brake and Accelerator Robot (CBAR) shown in Fig. 1.1. However, this product is very expensive. The high cost brings the need for a more cost effective solution that is adaptable from one car to another.

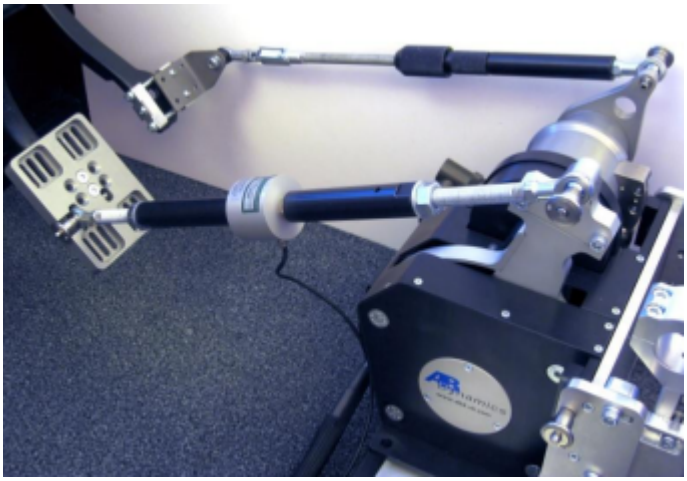

Fig. 1.1. Combined Brake and Accelerator Robot from AB Dynamics Ltd

## 1.2 Objective

The goal was to design and implement an actuator system to control the accelerator and brake pedal to accurately control the speed of an automobile. Two different control techniques were used and their performances were compared. This analysis also looks at the ability to control accurately from one car to another.

The first algorithm used was the very common PID controller. It is be compared to a Fuzzy controller. This paper also compares speed control with and without the use of a parallel linkage to control overshoot and steady state error in both algorithms.

A model of the plant is developed and used for simulation of the performance of the controllers. The plant includes a car model with a electric linear actuator and parallel linkage to push the accelerator and brake pedals. The two control algorithms are compared using simulation results as well as implemented and tested in two different cars.

The two cars being used are a 2007 Ford Focus and 2008 Toyota Prius. These cars were simply chosen because of their availability. The Ford Focus will be referred from here on as Car 1 and the Prius will be referred from here on as Car 2. The plant model is be based on Car 1. During the testing the controllers were tuned for Car 1.

Once reasonable results were achieved from both controllers the system was installed in Car 2 to evaluate the performance of the controllers.

# 2. PLANT MODEL

In this chapter the system will be modeled and simulated for the development of the control algorithms. There are two major components: linear actuator system and the car. The proceeding sections will take a closer look into both areas in order to model each subcomponent. At the end the individual transfer functions will be cascaded together to represent the plant in its entirety.

## 2.1 Car Model

A complete car model has many dynamic subsystems, but for the purpose of this paper a simplified model is used. Models for control are in general more simplified to help in designing the controller [3]. The focus will be on three subsystems: the accelerator pedal, engine, and car body.

### 2.1.1 Accelerator Pedal Model

The accelerator converts the actuator's linear position to an angle. From the accelerator pedal diagram in Figure 2.1, we have:

$$x_p = l_p \sin(\alpha) \tag{2.1}$$

The sine function causes this equation to be nonlinear. However, since the accelerator pedal does not rotate more than approximately twenty degrees, the small angle approximation can be taken advantage of [4]. This concept states that for small angles $\alpha$

$$\sin(\alpha) \approx \alpha \tag{2.2}$$

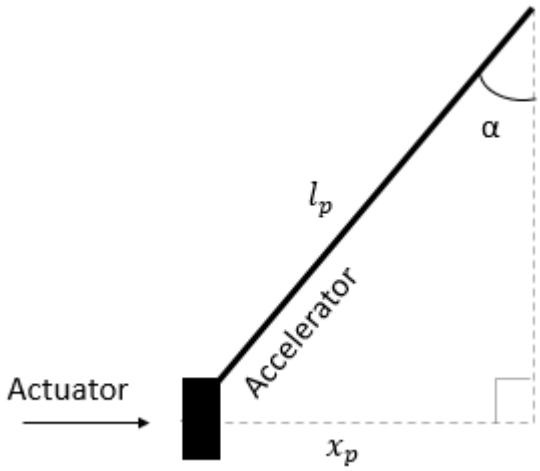


Fig. 2.1. Accelerator Pedal Diagram

and so the a new equation for $x_p$ can be written as:

$$x_p = l_p \alpha \tag{2.3}$$

which is now a linear equation. Using the Laplace transform and rearranging produces the transfer function for the accelerator pedal:

$$\frac{\alpha(s)}{X_p(s)} = \frac{1}{l_p} \tag{2.4}$$

Since $l_p$ is a constant it will be represented as a simple gain in the overall model.

### 2.1.2 Car Body Model

A momentum balance is one way to model a car body [5]. The major factor for momentum is the product of the velocity v and the mass m of the car. There are also momenta from the rotation of the crank shaft in the engine and the velocities of the cylinders, but these are much smaller than $mv$ and so will be ignored. $\theta$ is the slope of the road. The momentum balance can be written as:

$$m\frac{dv}{dt} + F_d = F_e - mg\sin(\theta) \tag{2.5}$$

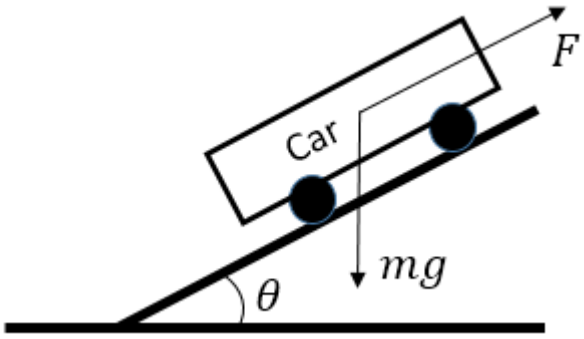


Fig. 2.2. Schematic Diagram of a Car on a Sloped Road [5]

where the term $F_d$ describes the momentum loss due to air resistance called the drag force [6]

$$F_d = \frac{1}{2}\rho v^2 c_D A \tag{2.6}$$

The the square of the velocity introduces a non-linearity to the model. To avoid this, it was assumed that:

$$v^2 \approx 2v \tag{2.7}$$

hence:

$$F_d \approx \rho v c_D A \tag{2.8}$$

Now let:

$$F_d = c_a v \tag{2.9}$$

where $c_a$ defined to be:

$$c_a = \rho c_D A \tag{2.10}$$

where $\rho$ is air density, A is the cross sectional area of the front of the car, and $c_D$ is the drag coefficient. The rolling resistance is small enough to be ignored [3]. $F_e$ is the force generated by the engine. The opposing force from the slope of the road is assumed to be proportional to the sine of the angle. However for this paper, it can be assumed that $\theta$ is zero because the testing of the PAEB takes place on a level track.

Also, it is assumed that $F_e$ is proportional to the signal $u$ sent to the throttle [5]. Hence, the momentum balance equation 2.5, becomes:

$$m\frac{dv}{dt} + c_a v = F_e \tag{2.11}$$

Now taking the Laplace transform:

$$msV(s) + c_a V(s) = F_e(s) \tag{2.12}$$

and rearranging produces the momentum balance car body model with engine force input and car velocity output:

$$\frac{V(s)}{F_e(s)} = \frac{1}{ms + c_a} \tag{2.13}$$

If it is assumed that $\rho = 1.225$ $(15C^\circ)$, $c_D = .4$, and $A = 3m^2$ (see Appendix A.2), the equation becomes:

$$\frac{V(s)}{F_e(s)} = \frac{1}{1200s + 1.47} \tag{2.14}$$

This model is compared to Simulink's $Simscape^{TM}$ $Driveline^{TM}$ full car model called `sdl_car`. Fig. 2.3 shows the open loop response of both models and it can be seen that the momentum car body model in equation 2.14 is not adequate. The momentum car body's response is very flat and that is due to it very large time constant. Because the

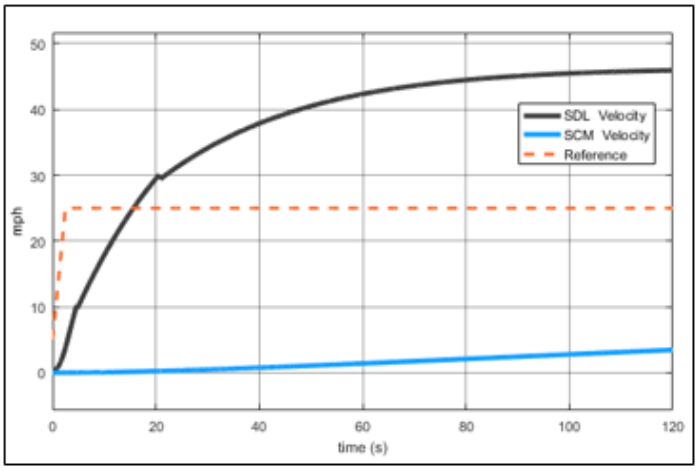


Fig. 2.3. Momentum Balance Model vs. MATLAB's SDL Car Body Model

momentum balance car model is not comparable to the `sdl_car` model, the `sdl_car` was used instead. It is more complex but it provides a more better representation of vehicle dynamics. It is shown in Fig 2.4. Notice this model takes into account the tires and brakes.

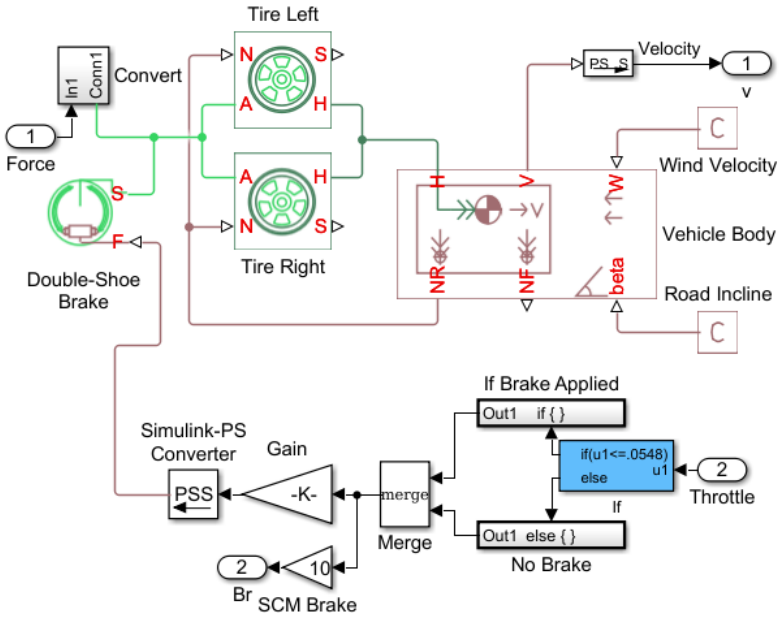


Fig. 2.4. MATLAB's SDL Car Body

### 2.1.3 Engine Model

There are many different engine models but the model in equation 2.15 was chosen from the Automotive Control Systems textbook because of its simplicity [3].

$$\frac{F_e(s)}{\alpha(s)} = \frac{k}{\tau s + 1} \tag{2.15}$$

When the accelerator angle changes the force $F_e(s)$ is not instantaneous, but instead, there is a time constant $\tau$, that represents the lag time before the engine reaches $F_e(s)$. The variable k, is depended on the speed of the car. There is also a dead time in the engine that takes place during the combustion process but for the sake of simplicity it is ignored.

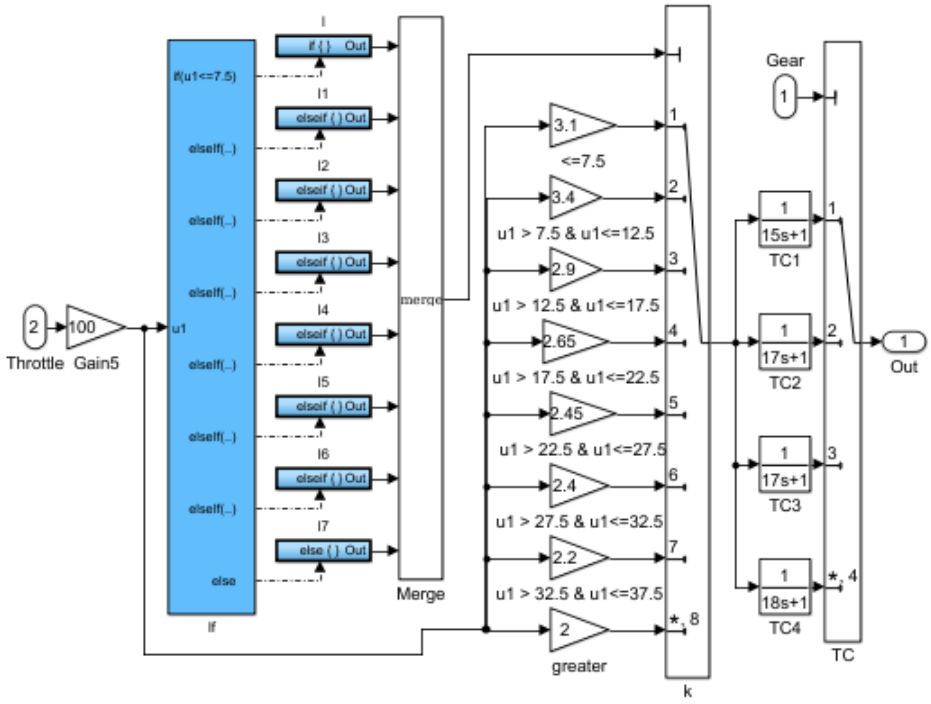


Fig. 2.5. Simplified Engine Model

The simplified engine model can be seen in Fig. 2.5. The gain, $k$, used in the engine is dependent on the throttle position. The engine time constant,$\tau$, is dependent on the current gear. The engine time constants and the gain k, were found by comparing the velocity output to that of the full `sdl_car` model.

### 2.1.4 Car Model

The engine model is combined with the vehicle body model to form the Simplified Car Model (SCM). It is shown in Fig. 2.6. The full `sdl_car` model is compared to this model to insure it is reasonable. Notice that the SDL Car model includes torque converter and transmission whereas the SCM omits those features for simplicity. However, the results are comparable. Fig. 2.7 shows that for reference of 25 miles the velocity curve is very similar and the gear numbers are very similar as well.

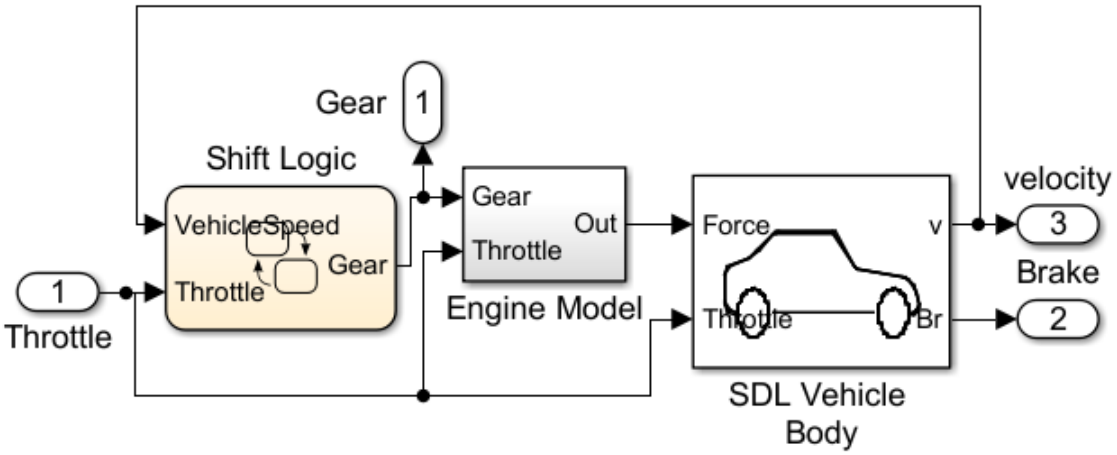


Fig. 2.6. Simplified Car Model (SCM)

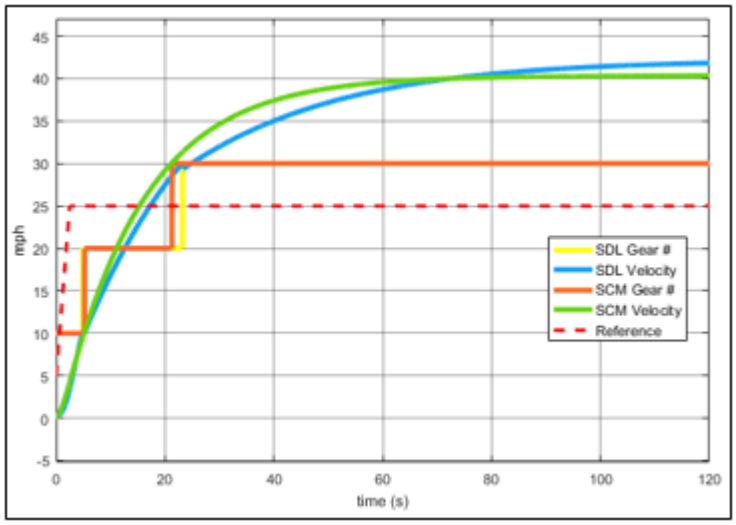


Fig. 2.7. SDL vs SCM Open Loop Simulation

## 2.2 Electric Linear Actuator Model

The electric linear actuator is an electromechanical system that converts voltage into linear movement. It will be used to control the position of the accelerator and the brake pedal. This actuator can be broken down into two components: DC Motor and the Ball-Screw.

### 2.2.1 DC Motor

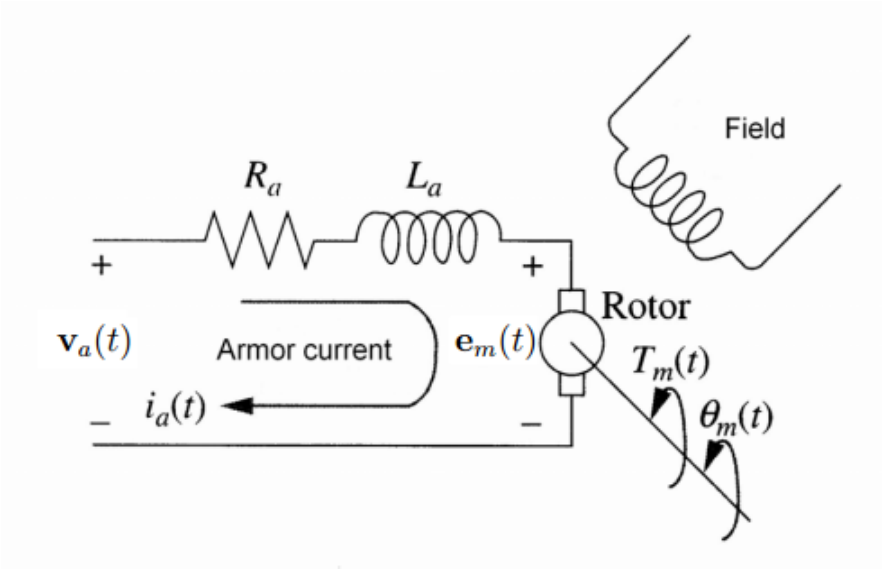


Fig. 2.8. DC Motor Circuit [7]

Figure 2.8 shows the electric diagram of a simple DC motor. In this figure, $v_a(t)$ is the applied armature voltage; it is the motor input. $R_a$ and $L_a$ are the armature resistance and inductance respectively. The voltage $e_m(t)$ is the back EMF created from the motion of the armature coil in the motor's fixed magnetic field. The back EMF voltage equation can be written as: [7] [8]

$$\mathbf{e}_m(t) = k_b \frac{d\theta}{dt} \tag{2.16}$$

where $k_b$ is the fem constant, $\theta$ is the angle of the motor shaft and $d\theta_m(t)/dt$ is the angular velocity of the motor. The mesh equation from the armature circuit in Figure 2.8 can be written as:

$$R_a i_a(t) + L_a \frac{di_a(t)}{dt} + e_m(t) = v_a(t) \tag{2.17}$$

Combining equations 2.16 and 2.17 produces:

$$R_a i_a(t) + L_a \frac{di_a(t)}{dt} + k_b \frac{d\theta}{dt} = v_a(t) \tag{2.18}$$

The Laplace transform yields:

$$V_a(s) + R_a I_a(s) + L_a s I_a(s) = k_b s \Theta(s) \tag{2.19}$$

and solving for: $I_a(s)$

$$I_a(s) = \frac{E_a(s) - k_b s \Theta(s)}{s L_a(s) + R_a} \tag{2.20}$$

The equation for the torque produced by the armature current is:

$$T_m(t) = k_\tau i_a(t) \tag{2.21}$$

where $k_\tau$ is the motor torque constant. The Laplace transform yields:

$$T_m(s) = k_\tau I_a(s) \tag{2.22}$$

Figure 2.9 shows the mechanical load on a motor. $J_m$ is the equivalent inertia in

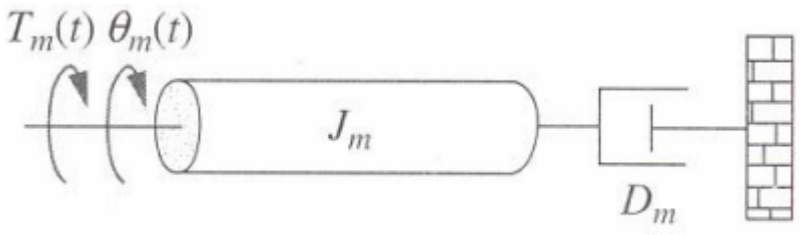


Fig. 2.9. Mechanical Load of a Motor

the armature and includes armature and load inertias. $D_m$ is the equivalent viscous damping in the armature. $T_m$ can be written as: [7] [8]

$$T_m(t) = J_m \frac{d^2\theta}{dt^2} + D_m \frac{d\theta}{dt} \tag{2.23}$$

where $K_\tau$ is the torque constant (in a consistent set of units the value of $K_\tau$ is equal to the value of $K_b$) and $J_m$ is equal to:

$$J_m = J_a + J_l(N1/N2)^2 \tag{2.24}$$

and $D_m$ is:

$$D_m = D_a + D_l(N1/N2)^2 \tag{2.25}$$

For simplicity it is assumed that the inertia and the damping effect from the load is minimal, resulting in:

$$J_m = J_a \tag{2.26}$$

and,

$$D_m = D_a \tag{2.27}$$

The Laplace transform of the equation 2.23 is:

$$T_m(s) = J_m s^2 \Theta(s) + D_m s \Theta(s) \tag{2.28}$$

Finally, combining equations 2.20, 2.22, 2.28 produces the DC motor transfer function:

$$\frac{\Theta_m(s)}{V_a(s)} = \frac{k_\tau}{J_m L_a s^3 + (D_m L_a + J_m R_a)s^2 + (D_m R_a + k_\tau k_b)s} \tag{2.29}$$

### 2.2.2 Angular to Linear Displacement

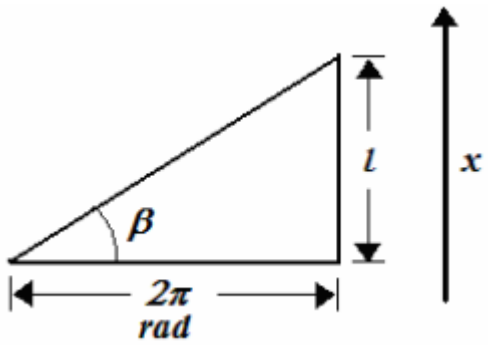


Fig. 2.10. Angular and Linear Movement Relationship

Figure 2.10 shows the relationship between the angular position of the motor and the linear advance from the ball-screw. $\beta$ represents the angle of the ball-screw lead, $l$ represents the step size of the lead after one revolution of the motor, and x(t) is the linear advance. Equation 2.30 represents the relationship between the angular position of the motor, $\theta(t)$, in radians, and linear advance of the actuator shaft $x(t)$ [8]

$$x(t) = \frac{l}{2\pi}\theta_m(t) \tag{2.30}$$

rewriting and taking the Laplace transform produces:

$$\theta_m(s) = X(s)\frac{2\pi}{l} \tag{2.31}$$

Substituting equation 2.31 into the DC motor transfer function results in:

$$\frac{X(s)\frac{2\pi}{l}}{V_a(s)} = \frac{k_\tau}{J_m L_a s^3 + (D_m L_a + J_m R_a)s^2 + (D_m R_a + k_\tau k_a)s} \tag{2.32}$$

Finally, equation 2.33 shows the combined transfer function of a linear actuator:

$$\frac{X(s)}{V_a(s)} = \frac{lk_\tau}{2\pi[J_m L_a s^3 + (D_m L_a + J_m R_a)s^2 + (D_m R_a + k_\tau k_a)s]} \tag{2.33}$$

### 2.2.3 Actuator Controller

The actuator model with the controller can be seen in Fig. 2.11. The parameters used in the actuator model were based on Tolomatic's ICR SmartActuator (config: ICR20S BN05 SM10 LMI SV1P CPS CNC1 MET FFG). The parameters can found in the Appendix A.1. In reality an actuator can only extent to the length of its shaft. To represent this physical limitation a saturation function was added to the model.

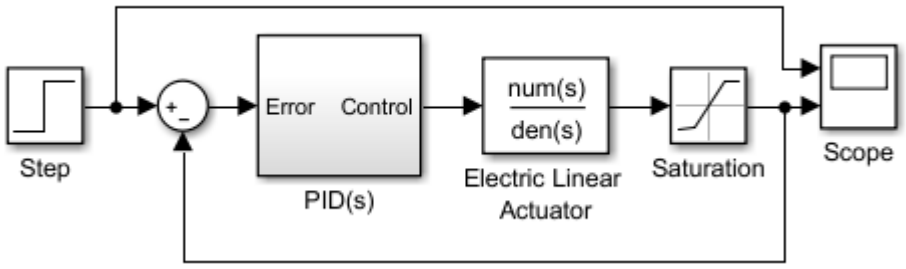


Fig. 2.11. Actuator Model

Simulink's PID Tuner was used to create a PID controller that emulates the SmartActuator's actual controller. The controller was designed to have a similar response to the actuator's actual response to a full-extend command. This simulation can be seen in Fig. 2.12.

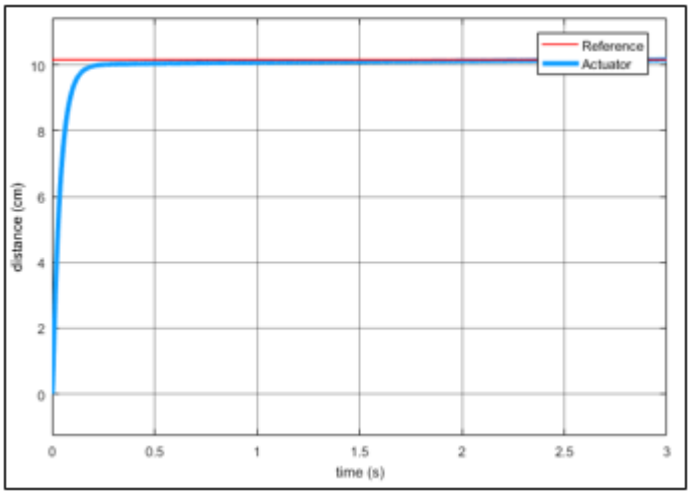


Fig. 2.12. Actuator Model Simulation

## 2.3 Parallel Linkage Design

During the initial testing it was seen that overshoot and steady state error was a problem. A braking action, in theory, would help decrease this problem. Every time the car surpassed the set point the brakes would be applied to slow down the car to the desired speed. A braking action can be achieved by adding second actuator to the system to apply the brakes. However, aside from the added design cost, there is the problem of controlling the two drives in such a way to avoid the undesired event where both the actuators are extended at the same time; the brake and the accelerator are being applied at the same time.

To combat this issue, a cost effective design known as the Four-par parallel linkage was used. This is a subset of Watt's Linage. There are two sets of parallel linkage to form a parallelogram. An illustration can be seen in Fig. 2.13 and Fig. 2.14. The black dots represents joints that allow the linages to swing. The orange dots are secured joints. The semi vertical linages pivot about the orange points. The vertical position of orange dots dictates the amount of movement the brake linage travels

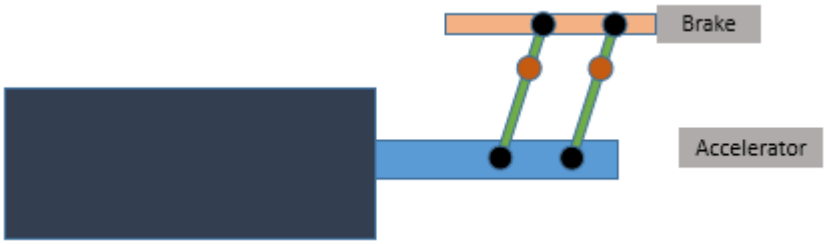


Fig. 2.13. Parallel Linkage: Brake Applied

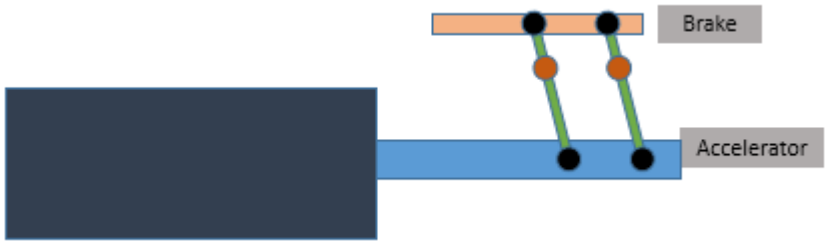


Fig. 2.14. Parallel Linkage: Accelerator Applied

The braking action is modeled in the vehicle body model in Fig. 2.4. The logic is such that the brake is applied when the actuator is retracted. The brakes will be applied until the car slows down to the set point.

## 2.4 Plant Model

The engine and vehicle body are combined with actuator model to form the complete plant. The diagram can be seen in Fig. 2.15. The open loop simulation of the plant is very similar to that of the SCM as shown in Fig. 2.16.

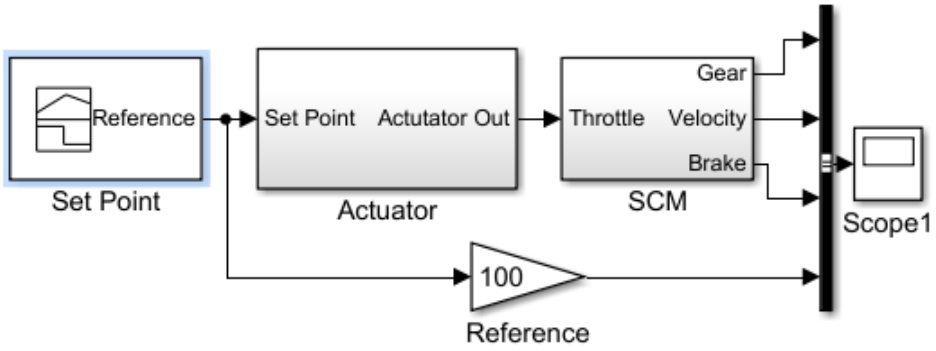


Fig. 2.15. Plant

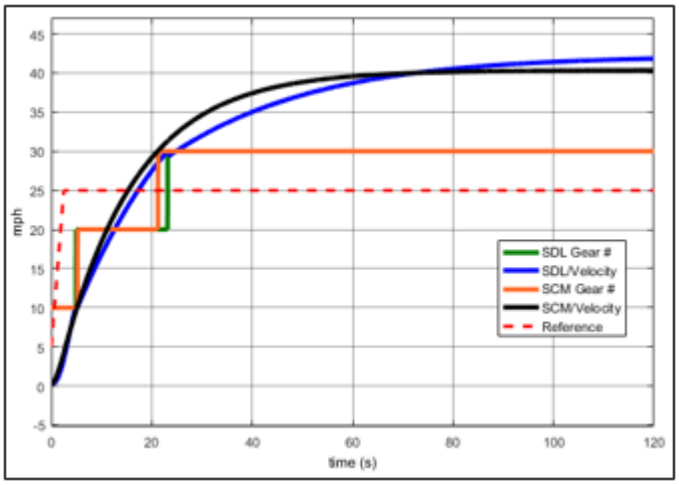


Fig. 2.16. Open Loop Plant vs. SDL Car Simulation

# 3. PID CONTROL

## 3.1 PID Design

The first control strategy covered is the industry common PID controller. Carl Johan Astrom defines the PID control law as:

$$u(t) = Ke(t) + \frac{K}{T_i}\int_0^t e(\tau)\,d\tau + KT_d\frac{de(t)}{dt} \tag{3.1}$$

where:

$$e(t) = r(t) - y(t) \tag{3.2}$$

where y is the measured plant output and r the reference. The controller parameters are proportional gain $K$, integral time $T_i$, and derivative time $T_d$. It is desirable for the integration to go to infinity and the derivative time go to zero [5]. The control law is a sum of three terms: the proportional term that is proportional to the error, the integrating term that is proportional to the integral of the error, and a derivative term that is proportional to the derivative of the error. The three terms are a control strategy based on the past, the present and the future. The integral term contains past information about the system behavior, the proportional term correlates to the present behavior and the derivative term is a prediction of how the system will behave in the future.

Now by letting:

$$K_p = K, K_i = \frac{K}{T_i}, K_d = KT_d \tag{3.3}$$

produces the classical derivation of PID control [9].

$$u(t) = K_p e(t) + K_i\int_0^t e(\tau)\,d\tau + K_d\frac{de(t)}{dt} \tag{3.4}$$

A strict implementation of the three term PID control law will most likely not result in a good controller [5]. It is well known that differentiation is sensitive to noise. A

simple example of this be seen in the example below. The output signal y, contains a plant modeled as $sin(t)$, with added noise.

$$y(t) = sin(t) + noise(t) = sin(t) + a_n sin(\omega_n t) \tag{3.5}$$

The derivative of the output signal is:

$$\frac{dy(t)}{dt} = cos(t) + a_n \omega cos(\omega_n t) \tag{3.6}$$

Notice the signal to noise ratio for the output signal is $\frac{1}{a_n}$ but the signal to noise ratio of the differentiated output signal is $\frac{\omega}{a_n}$. This ratio can be very high if $\omega$ is large. In practice it is often necessary to filter the high frequency gain of the derivative term. This can be accomplished by altering the derivative term to:

$$D(s) = K_d \frac{N}{1 + N\frac{1}{s}} e(s) \tag{3.7}$$

where N is the pole location of the filter in the derivative action. The new PID control law becomes:

$$u(s) = K_p e(s) + \frac{K_i}{s} e(s) + K_d \frac{N}{1 + N\frac{1}{s}} e(s) \tag{3.8}$$

## 3.2 PID Control Simulation

This new controller is represented in Simulink in Fig. 3.2. Simulink's PID Tuner was used to help tune in the plant. Fig. 3.3 shows the simulation results for 10 mph, 25 mph, and 45 mph. The overshoot is near 10% for all three speeds. The settling time is higher than desired: 20, 27, and 30 seconds respectively. There is no steady state error for the 10 and 25 mph simulation with the PID controller. The 40 mph simulation has some error but it is very small: .3 mph or .75%. Finally, notice the brakes were only applied by the controller in the 10 mph simulation.

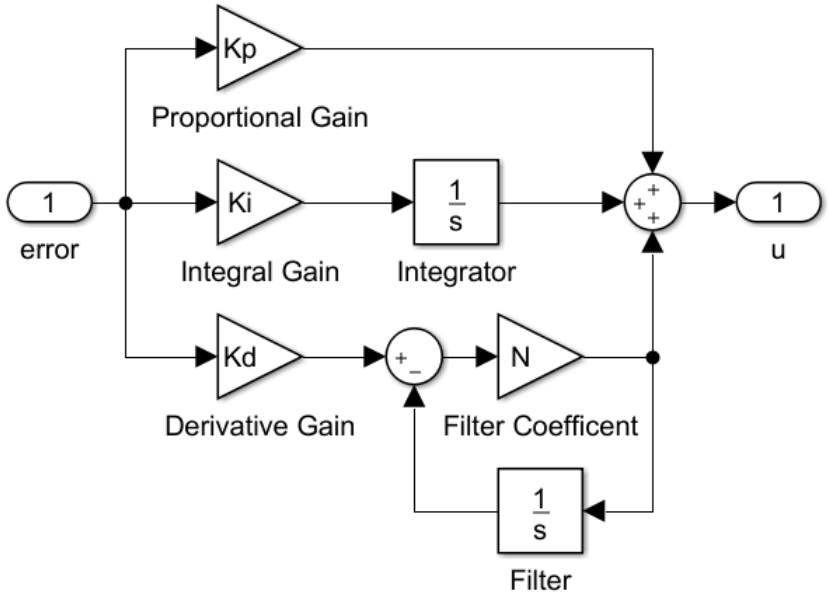


Fig. 3.1. PID Simulink Model with Derivative Filter

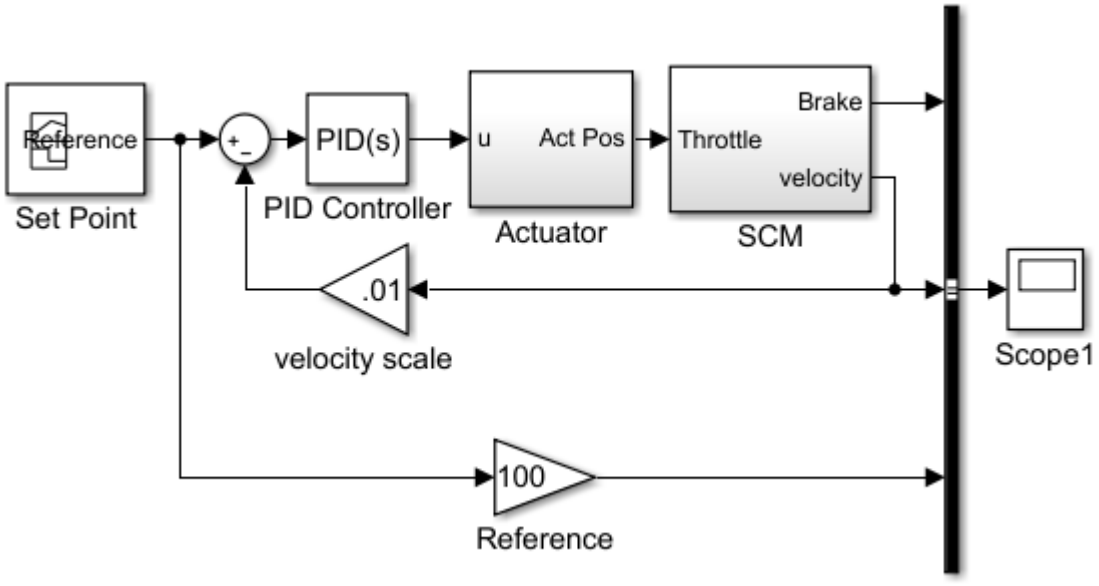


Fig. 3.2. PID Simulink Model with Plant

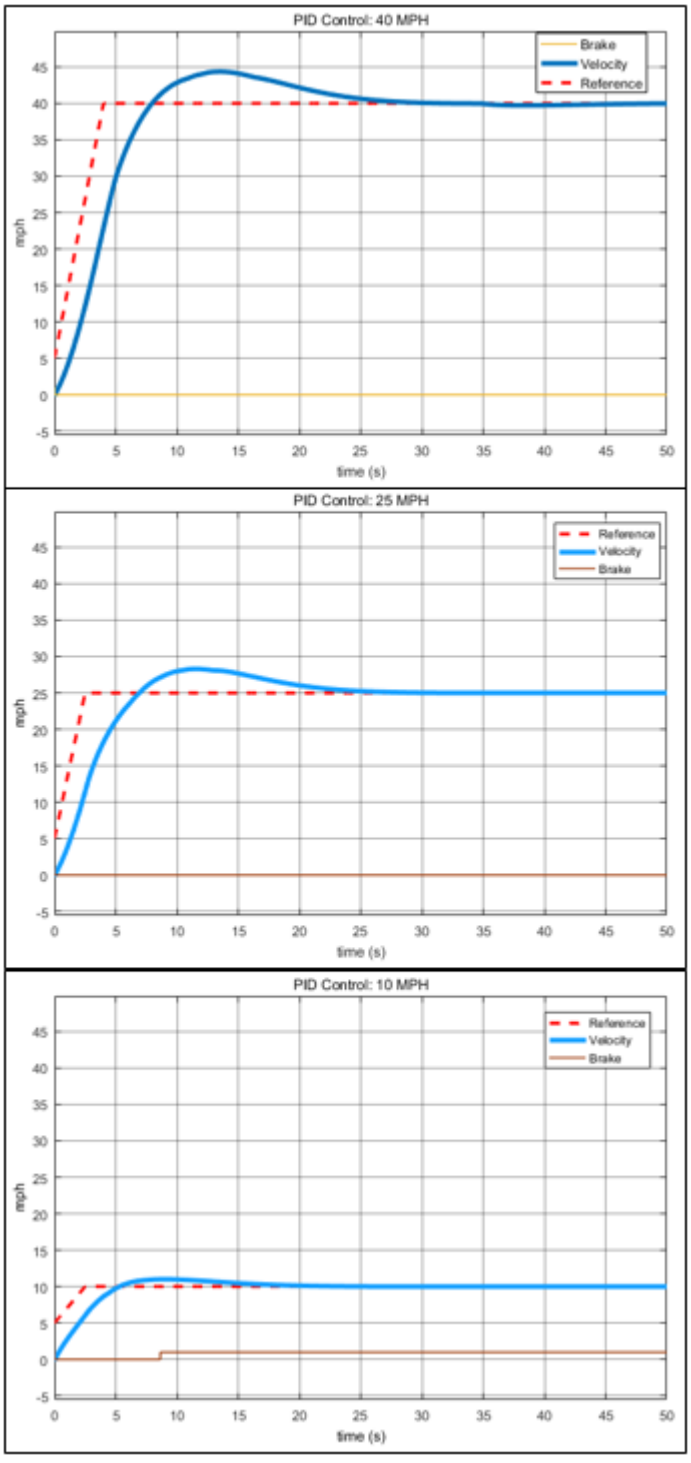


Fig. 3.3. PID Simulation Results

### 3.3 Implementation

The Implementation of a continuous-time PID controller in a digital computer requires the approximation of the derivatives and the integrals in the control law. One possible way to do this is shown in the following equations. The proportional term does not change:

$$P(n) = K_p e(n) \tag{3.9}$$

where:

$$e(n) = r(n) - y(n) \tag{3.10}$$

The integral action:

$$I(t) = K_i \int_0^t e(\tau)\, d\tau \tag{3.11}$$

is converted to a derivative. The derivative can then be estimated as a difference.

$$\frac{dI(t)}{dt} = K_i e(t) \tag{3.12}$$

$$\frac{I(n+1) - I(n)}{T_s} = K_i e(n) \tag{3.13}$$

moving the $T_s$ to the other side and solving for the future term produces the discretized integral equation:

$$I(n+1) = I(n) + K_i T_s e(n) \tag{3.14}$$

The modified derivative term from equation 3.7 can be written as:

$$sD(s) + ND(s) = K_d N s e(s) \tag{3.15}$$

Using the inverse Laplace transform to the convert to time domain:

$$\frac{dD(t)}{dt} + ND(t) = K_d N \frac{de(t)}{dt} \tag{3.16}$$

and substituting the derivative term with a difference equation:

$$\frac{D(n) - D(n-1)}{T_s} + ND(n) = K_d N \frac{e(n) - e(n-1)}{T_s} \tag{3.17}$$

and rewriting produces the discretized derivative term with a high frequency filter.

$$D(n) = \frac{K_d N T_s}{(1 + N T_s)}[e(n) - e(n-1)] + \frac{1}{(1 + N T_s)} D(n-1) \tag{3.18}$$

Tying the discretized equations together results in the equations below. It is one possible version of a digital implementation of a PID controller.

$e(n) = r(n) - y(n)$

$P(n) = K_p e(n)$

$D(n) = \frac{K_d N T_s}{(1+N T_s)}[e(n) - e(n-1)] + \frac{1}{(1+N T_s)} D(n-1)$

$u(n) = P(n) + I(n) + D(n)$

$I(n+1) = I(n) + K_i T_s e(n)$

# 4. FUZZY CONTROLLER

The second control strategy used is Fuzzy controller. This control strategy takes on a human thinking approach to controlling the plant. A set of if-then rules defines what the controller should do based on the input. Fuzzy control is based on four main parts: rule-base, inference mechanism, fuzzification interface, and defuzzification interface [10]. Fig. 4.1 shows the relationship between the four parts and the process. The inner workings of this controller are detailed in the following sections.

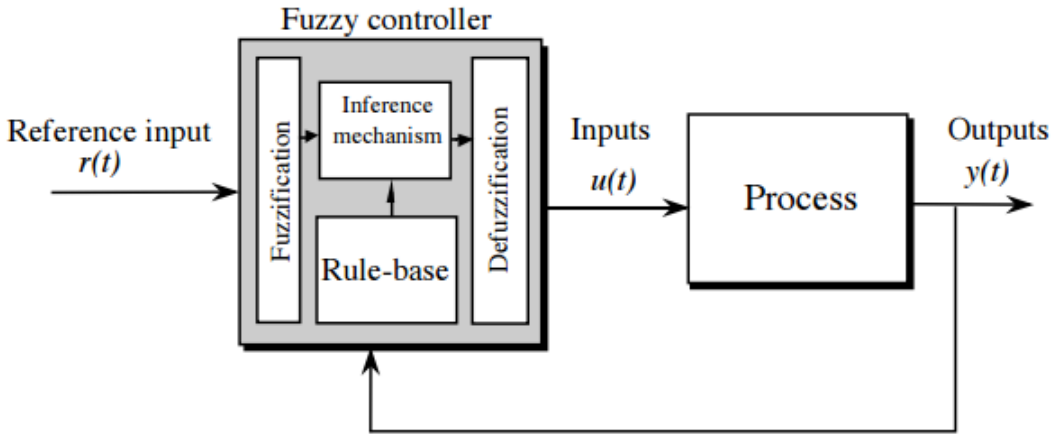


Fig. 4.1. Fuzzy Control Architecture [10]

## 4.1 Fuzzy Design

A proportional-derivative Fuzzy Controller was chosen because of the information the inputs provide. The first input is the error equation:

$$e(t) = r(t) - y(t) \tag{4.1}$$

where $y(t)$ is the automobile velocity and $r(t)$ is the reference signal. $r(t)$ is initially a ramp and then levels off at the set point speed. The second input is chosen to be the negative derivative of the velocity. The derivative of the error equation is:

$$\frac{de}{dt} = \frac{dr}{dt} - \frac{dy}{dt} \tag{4.2}$$

however, after the ramp plateaus, $r(t)$ becomes constant. At that point the derivative of $r(t)$ becomes zero. Resulting in the second input to the fuzzy controller: the negative derivative of the velocity.

$$\frac{de}{dt} = -\frac{dy}{dt} \tag{4.3}$$

The two inputs are valuable because it gives information about what the car is currently doing and what it will do in the future. The proportional-derivative fuzzy controller is depicted in Fig. 4.2.

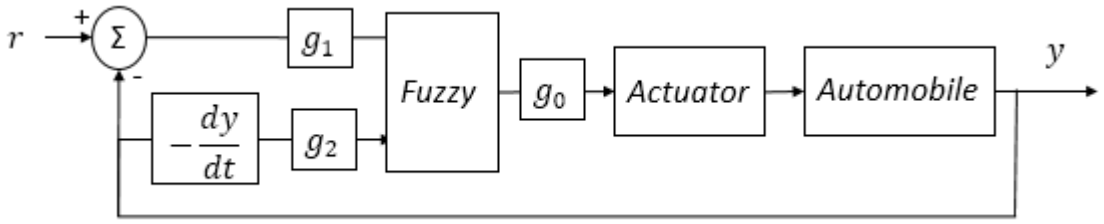


Fig. 4.2. PD Fuzzy Controller Block Diagram [11]

With the inputs defined, a rule-base can be created. A rule-base is a set of If-Then rules, linguistic representation, of how to achieve good control. The if-then rules of three extreme events for the two inputs are described in the list below ($-\Delta$V is the negative derivative of the velocity).

- If (Error is VN) and ($-\Delta$V is VN) then (Output is LD): If the error is very negative (the car is going very fast) and the change in velocity ($-\Delta$V) is increasing very quickly, then decrease the speed by a large amount (LD).

- If (Error is ZE and ($-\Delta$V is ZE) then (Output is NC): If the error is zero (the car is at the set point velocity) and the change in velocity ($\Delta$E) is not increasing or decreasing (constant), then make no change (NC) to the controller output.

- If (Error is VP) and ($-\Delta$V is VP) then (Output is LI): If the car is going very slowly (VP) and the change in velocity ($\Delta$E) is very positive (the car is slowing down at large rate), then increase the output my large amount (LI).

The complete rule-base can be seen in Table 4.1. The acronyms are defined in Table 4.2. The extreme cases are obvious but the intermediate rules appear subjective. However, these rules were adjusted in the tuning phase to achieve the desired performance.

Table 4.1.
Rule-Base

| u | | Error | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | VN | MN | SN | ZE | SP | MP | VP |
| -ΔV | VN | LD | LD | LD | LD | MD | SD | NC |
| | MN | LD | LD | MD | MD | SD | NC | SI |
| | SN | LD | MD | SD | SD | NC | SI | MI |
| | ZE | LD | MD | SD | NC | SI | MI | LI |
| | SP | MD | SD | NC | SI | SI | MI | LI |
| | MP | SD | NC | SI | MI | MI | LI | LI |
| | VP | NC | SI | MI | LI | LI | LI | LI |

Table 4.2.
Rule-Base Acronym Table

| Input | | Output | |
|---|---|---|---|
| VN | Very Negative | LD | Large Decrease |
| MN | Medium Negative | MD | Medium Decrease |
| SN | Slightly Negative | SD | Slight Decrease |
| ZE | Zero | NC | No Change |
| SP | Slightly Positive | SI | Slight Increase |
| MP | Medium Positive | MI | Medium Increase |
| VP | Very Positive | LI | Large Increase |

This leads to the membership functions that are needed in the fuzzification, inference, and finally defuzzification process. Fig. 4.3 shows a generic triangular membership functions for the input and the output. B is the maximum value of the controller output and A is the input that results in an output of B. The gains $g_0$, $g_1$, and $g_2$ Fig. 4.2 can be adjusted to map their respective signals to the bounds of A and B. 4.4, Fig. 4.5, and Fig. 4.6 show the two inputs and output membership functions designed in Matlab. Notice that these membership function have different widths. That also results from tuning to obtain desirable results from the controller.

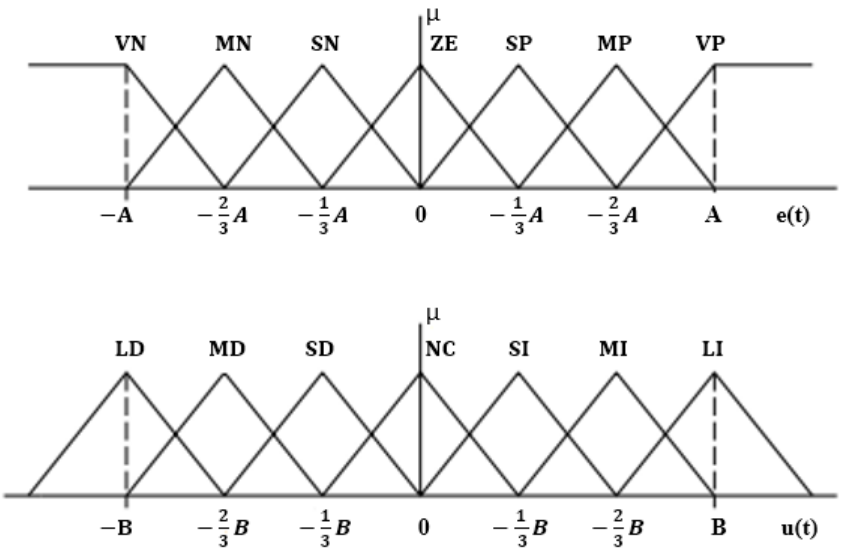


Fig. 4.3. Fuzzy Membership Functions [12]

The whole process from crisp input to linguistic input to linguistic output to crisp out is a complex process but Fig. 4.7 does a great job of depicting the process. $T_0$ and $P_0$ are the two inputs. This figure shows how the two crisp inputs are made fuzzy by converting them to linguistic variables with membership grades between 0 and 1. This grade is found through linear extrapolation. The MIN-MAX inference scheme is used to determine the linguistic outputs $\mu_B 1$ and $\mu_B 2$ for the two rules. In general there are R $\mu_B i$ values, where R is the number of rules. The designed controller uses 49 rules (7 input members by 7 output members). The defuzzification process uses

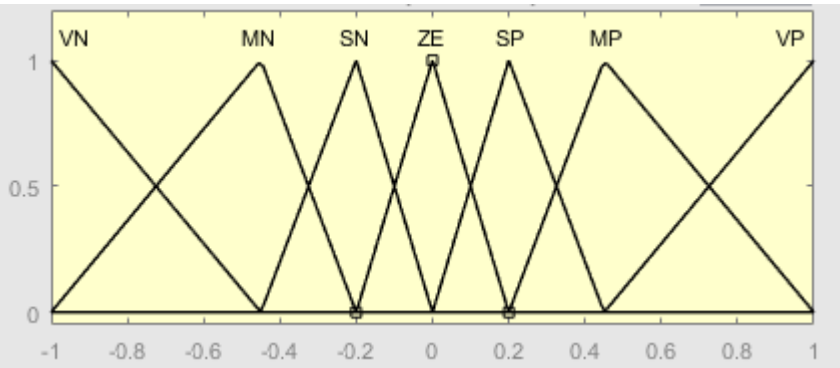


Fig. 4.4. Fuzzy Error Membership Function

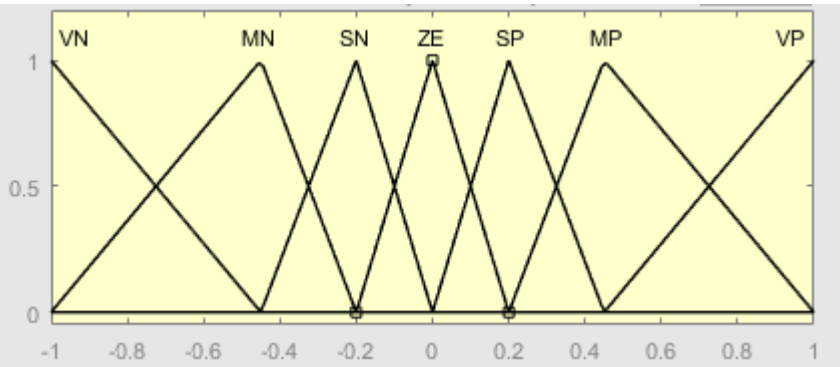


Fig. 4.5. Fuzzy Change-in-Velocity Membership Function

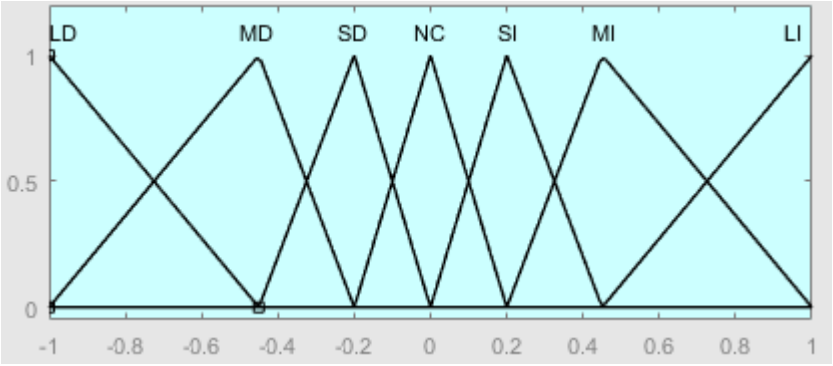


Fig. 4.6. Fuzzy Output Membership Function

center of gravity (COG) or centroid method to obtain a crisp output. This achieved by using equation 4.4

$$y_n^{crisp} = \frac{\sum_{i=1}^{R} b_i^n \int \mu_{Bi^n}(y_n) dy}{\sum_{i=1}^{R} \int \mu_{Bi^k}(y_n) dy} \tag{4.4}$$

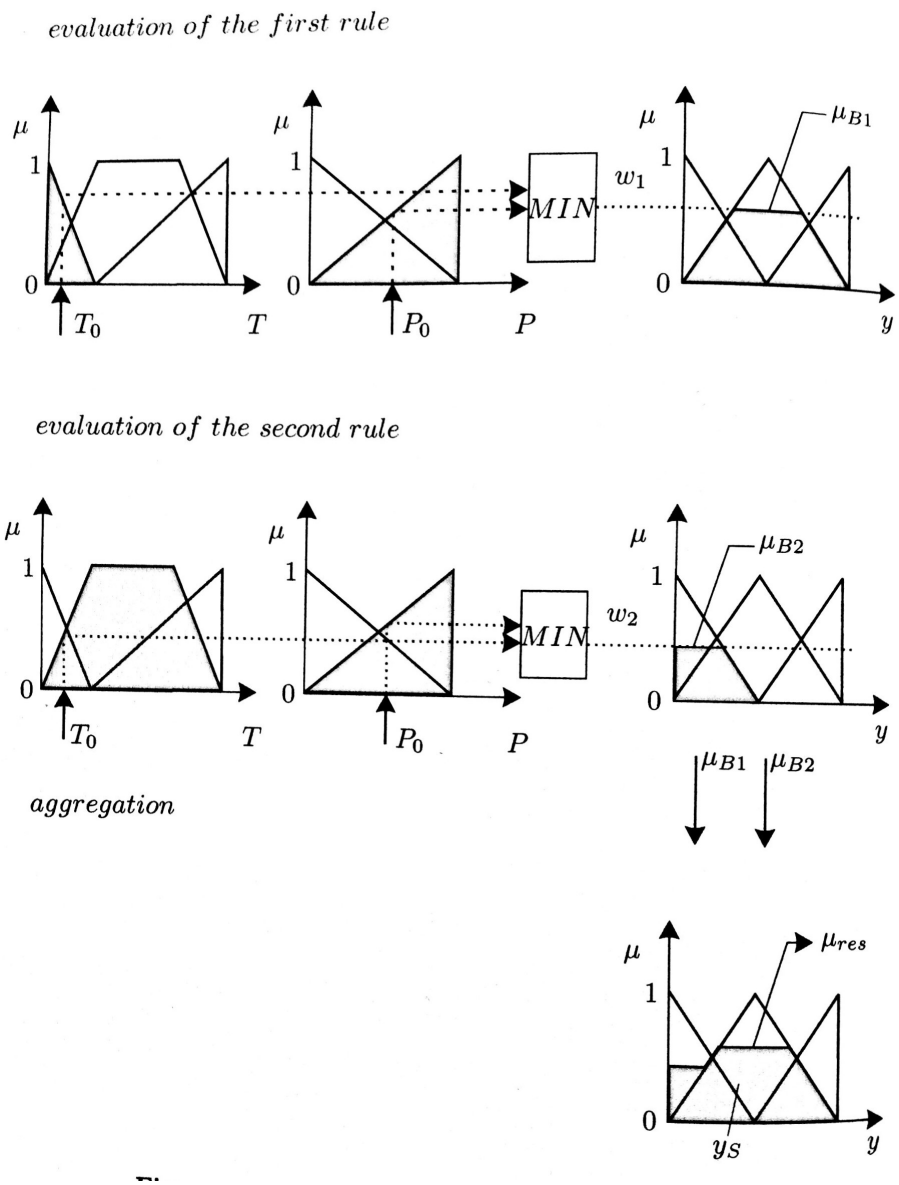


Fig. 4.7. Evaluation of Fuzzy Rule-Base [3]

where $y_n^{crisp}$ is the crisp output at discrete time n, $b_i^n$ is the center of the membership function $\mu_B i^n$ (the location of the peak of the membership function), and

$$\int \mu_{Bi^n} \tag{4.5}$$

is the area under the membership function $\mu_{Bi^n}$.

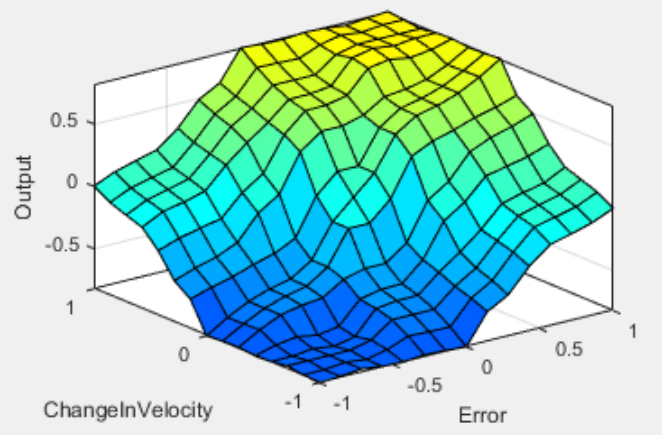


Fig. 4.8. Fuzzy Surface

Fig. 4.8 is a fuzzy surface that shows the relationship between the the two inputs and the output. A quick sanity check proves out. At the extreme input of [-1,-1] (Error, ChangeInVelocity), the output is -1 which is correct. When converted to linguistic terms: if the error and the velocity are very negative (VN) then from the Rule-Base in Table 4.2 the output should be large decrease (LD) or -1. The opposite is true about the inputs 1,1. Another check at (1,-1) also checks out: an error that is very positive (VP) and a change in velocity that is very negative (VN) results make no change (NC). Notice the symmetry of the Rule-Base surface matches the symmetry of the Rule-Base table.

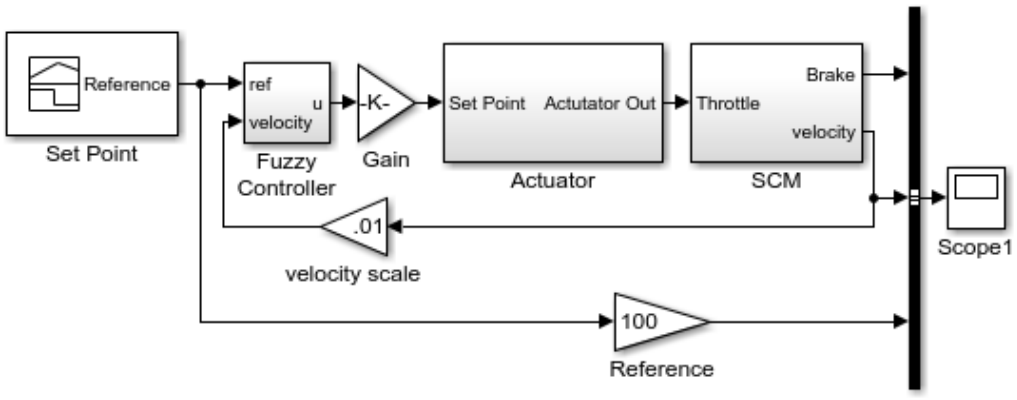


Fig. 4.9. Simulink Diagram of Fuzzy Controller with Plant

## 4.2 Fuzzy Control Simulation

Finally, Fig. 4.9 and Fig. 4.10 shows the Simulink model of the PD-Fuzzy controller with the plant. The simulation results are shown in Fig. 4.11. Unlike the

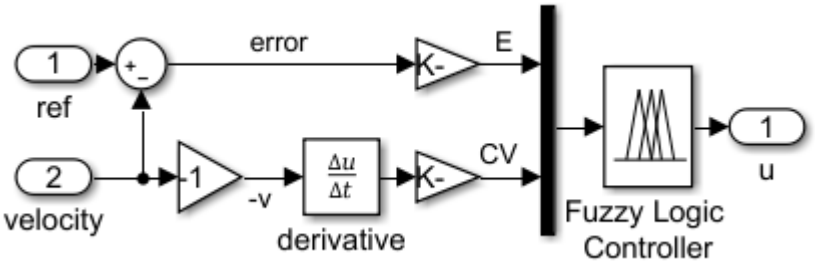


Fig. 4.10. PD Fuzzy Controller Simulink Diagram

PID simulation the Fuzzy controller response has no steady state error and has no overshoot. It also has a much faster settling time at the three chosen velocities. The Fuzzy controller has a settling time of 6, 8, 7 seconds for 10 mph, 25 mph, and 40 mph respectively while the PID simulation has 20, 27, 30 seconds respectively.

Similar to the PID simulation the brake was extensively used on the 10 mph simulation. This makes sense because the car is more likely to overshoot the reference on its own because of the low speed. The brake was also used briefly on the 40 mph simulation. The brakes were applied as the automobile was reaching the reference. It appears that brake usage helped eliminate overshoot.

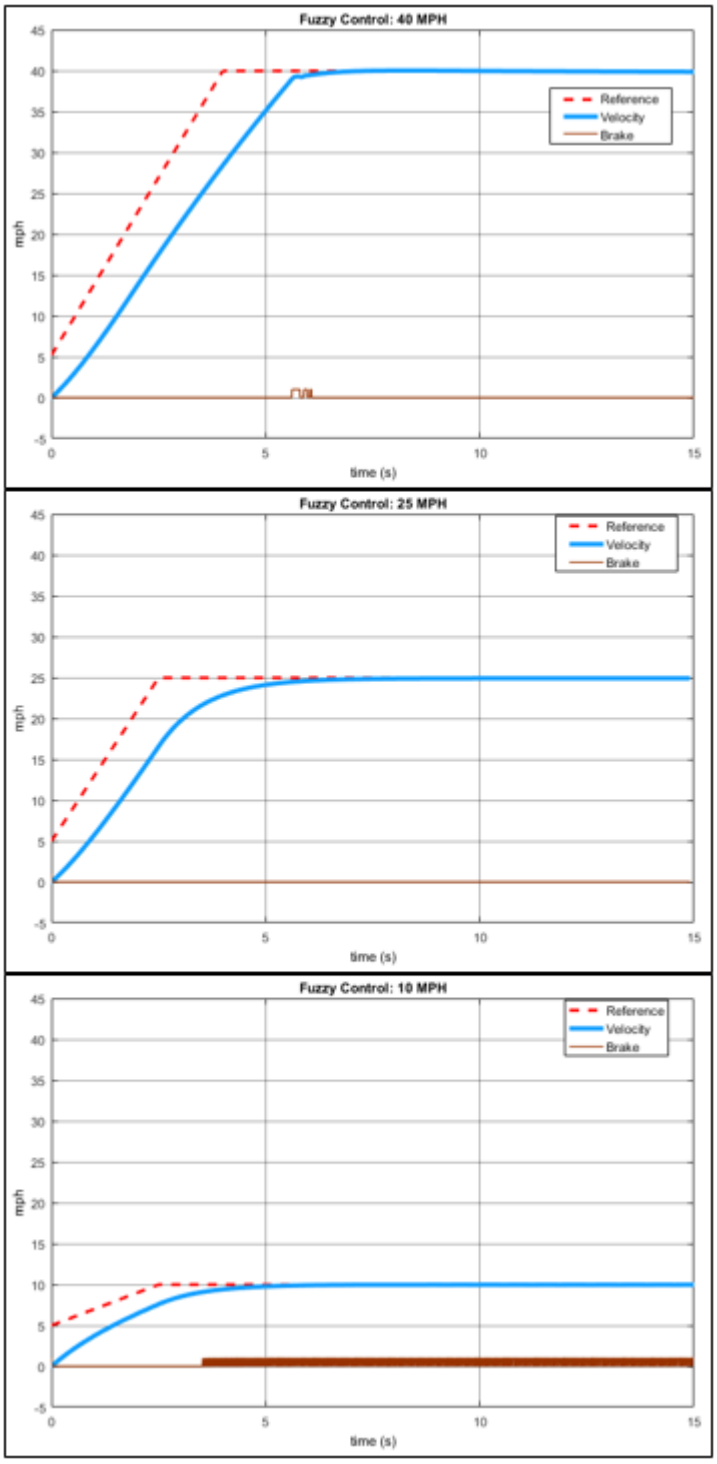


Fig. 4.11. Fuzzy Control Simulation

### 4.3 Implementation

Like the PID controller, this control law must to be discretized to work in a microcontroller. The error signal is a simple difference equation so no change is necessary.

$$e(n) = r(n) - y(n) \tag{4.6}$$

However, a pure derivative can not be realized in discrete time and so it must be approximated.

$$\frac{de}{dt} = -\frac{dy}{dt} \tag{4.7}$$

Like in the PID controller, the derivative can be approximated with a backwards difference equation: [13]

$$cy(n) = -\frac{y(n) - y(n-1)}{T_s} \tag{4.8}$$

Where $T_s$ is the sampling period. The microcontroller realization is:

$e(n) = r(n) - y(n)$

$cy(n) = -\frac{y(n)-y(n-1)}{T_s}$

$FuzzyIn(e(n), cy(n))$

$u = FuzzyOut()$

# 5. TESTING AND EXPERIMENTAL RESULTS

## 5.1 Implementation

After the simulations were completed both controllers were tested in Car 1 and Car 2. Before the testing could be completed, the hardware and software had to be built and developed respectively. Fig. 5.1 shows the electrical hardware system diagram. The system was powered from the cars' 12V auxiliary port. An Arduino

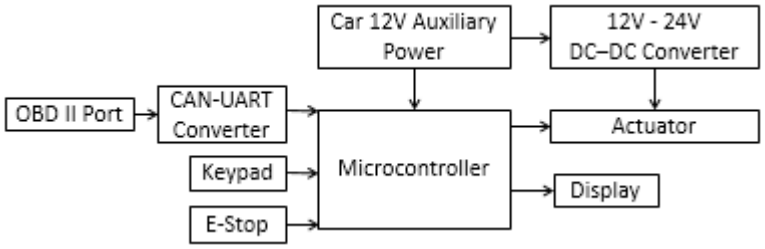


Fig. 5.1. Electrical Hardware System Diagram

Mega 2560 microcontroller was used because of its ability to accept a 12V power input. A 12V-24V DC-DC converter was used to step up the voltage to 24V to power the linear actuator. A 20x4 LED display provided system information to the user. A keypad was used for the actuator setup and to enter the desired set point. The velocity feedback was obtained through the cars' OBDII port. A CAN-UART converter converted the communication from the OBDII port to UART, a protocol that is supported on the arduino microcontroller. Lastly, an E-stop button was added to retract the actuator any time the process needed to be stopped. The software was developed in Arduino Software, Arduino's integrated development environment. The software flowchart is shown in Fig. 5.2 illustrates the software's major components. After initializing all variables the software goes through an actuator setup loop. The

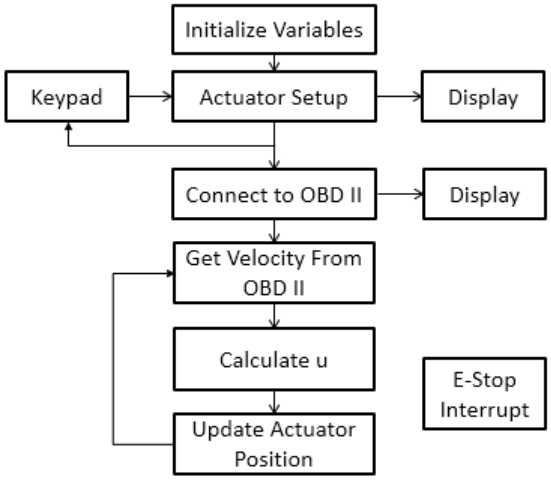


Fig. 5.2. Software Flowchart

actuator parameters and speed reference can be changed and set via the keypad. Once the setup is complete and the user starts the controller the software goes in a endless loop where it requests the speed of the car through the OBDII port, uses this speed to calculate the controller output, and finally the microcontroller updates the actuator position via digital pins. This process could be interrupted with the push of the E-stop button.

Fig. 5.3. System Installed in Car 1

The actuator system was built in a shop. The base was made out of pine boards and plywood. The parallel linkage consisted up angle iron for the linkages and bolts and washers for the joints. The orange joint in Fig. 2.13 was moved as close to the actuator as possible to obtain maximum travel distance of the brake linkage. This was done because distance between the accelerator and the brake pedal is small.

Fig. 5.3 shows the whole system installed in Car 1. The actuator system sits in the floorboard and is secured by the floorboard walls (packaging foam may need to be used to secure the actuator when the product is used in different cars because the floorboards vary for each make and model). The controller box, that houses all the electrical hardware, sits in the passenger seat. Fig. 5.4 provides a closer look of the controller box and the actuator system.

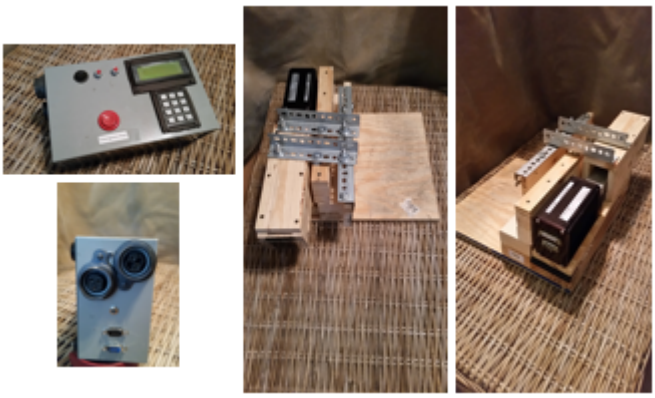

Fig. 5.4. Controller Box and Actuator System

## 5.2 PID Control Adjustments

The PID parameters from the simulation did not produce desirable results. This was in part predicted because the car model used in the simulation was simplified and didn't account for all the dynamic subsystems that effect the car speed. Thus the control parameters needed to be manually tuned for the Car 1 and Car 2. The parameters were systematically tuned by starting with the proportional gain parameter

$K_p$. The tuning was started by adjusting the proportional gain and keeping the other parameters zero. After finding the best value for $K_p$, another parameter was adjusted while keeping the parameters constant. Fig. 5.5 illustrates this process. Table. 5.5 was used to understand the basic effects each parameter had on the the system [14].

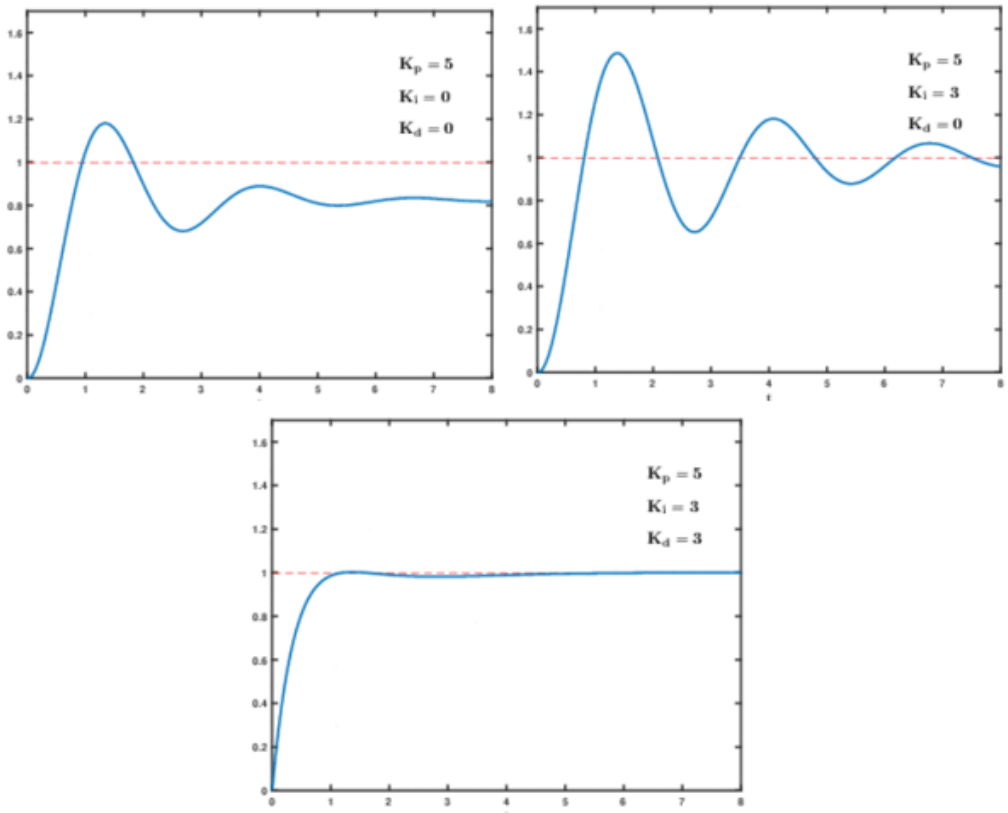


Fig. 5.5. PID Tuning Process Example

Table 5.1.
PID Parameter Effects

| **Params** | **Rise Time** | **Overshoot** | **Settling Time** | **SS Error** | **Stability** |
|---|---|---|---|---|---|
| $K_p$ | Decrease | Increase | Small Change | Decrease | Degrade |
| $K_i$ | Decrease | Increase | Increase | Eliminate | Degrade |
| $K_d$ | Minor Change | Decrease | Decrease | No Effect | Improve |

## 5.3 Fuzzy Control Adjustments

Unlike the PID controller, the Fuzzy controller had fewer adjustments that were needed. The controller output gain needed to be increased until the controller pushed the accelerator far enough. Again, this is from the simulation model not being exact. However, this took much less time than tuning the PID controller. In general this may not be the case because of the complexity of the fuzzy controller.

## 5.4 Actual Results Compared

The test matrix in Fig. 5.6 shows all the test combinations completed during the testing. WPL and NPL are acronyms that are defined as "With Parallel Linkage" and "No Parallel Linkage" respectively. Most tests was completed 3 times. This results in 3 x 24, or 72 tests.

| Controller | | | Speed | | |
|---|---|---|---|---|---|
| | | | 10 MPH | 25 MPH | 40 MPH |
| PID Control | Car 1 | NPL | | | |
| | | WPL | | | |
| | Car 2 | NPL | | | |
| | | WPL | | | |
| Fuzzy Control | Car 1 | NPL | | | |
| | | WPL | | | |
| | Car 2 | NPL | | | |
| | | WPL | | | |

Fig. 5.6. Test Matrix

### 5.4.1 Parallel Linkage Results

All the test scenarios were completed twice: once without the parallel linkage (NPL) and again with the parallel linkage (WPL). The comparison of the two sets of data found that there was not a significant difference in the performance of the

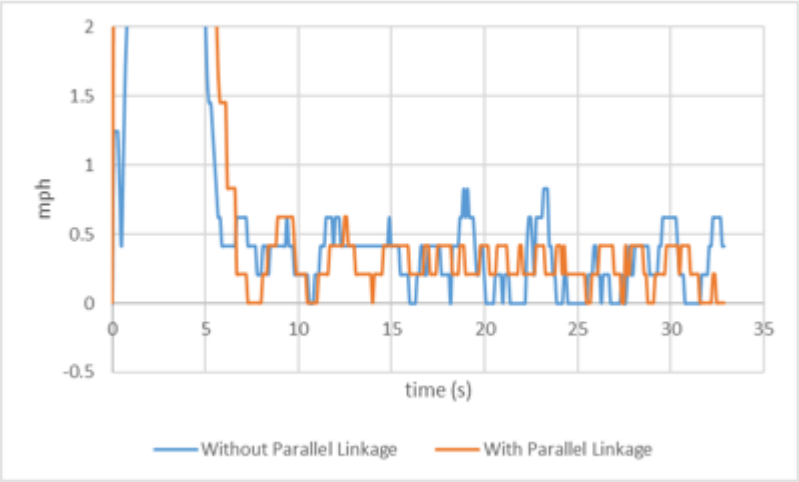


Fig. 5.7. With and Without Parallel Linkage Results

NPL and WPL in either car. Fig. 5.7 shows the difference of the averaged absolute errors of the Fuzzy controller in Car 1 at 10 mph. The absolute value of the error curves were used so that the error curves could not cancel each other out. The plot shows that for the fuzzy controller in Car at 10 mph the WPL did slightly better than the NPL. However, in some other scenarios the NPL did slightly better than WPL. Nevertheless, in all cases there was no significant difference. This fact can most likely be explained from the physical realization of the parallel linkage. From Fig. 5.8 the travel distance d, is the distance that the brake plunger travels during the braking action. If d is not very large than the brake pedal will not be decompressed very much, resulting in limited effect on the system. Ideally the green linkage should be lengthen and the orange stationary joints should be adjusted until a desirable d is achieved. However, given the small distance between the accelerator and the brake pedal, the distance of the green linkage was limited, resulting in a small d. To achieve a larger d a more advanced parallel linkage design will need to be used.

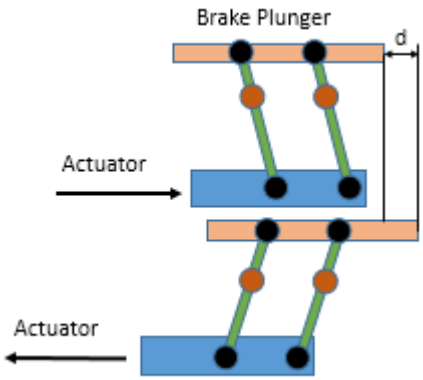


Fig. 5.8. Parallel Linkage Travel Distance

### 5.4.2 Fuzzy vs. PID

One of the main comparison of this paper is the Fuzzy vs. PID performance in Car 1. From the test matrix in Fig. 5.6 three car speeds were considered: 10 mph, 25 mph, and 40 mph. I choose to show the results for 25 mph in this chapter but all the results can be found in Appendix B. The top two plots of Fig. 5.9 show the PID and Fuzzy controller results for Car 1 with a reference of 25 mph. The bottom plot of Fig. 5.9 shows a zoomed in error curve of the Fuzzy controller. It can be seen that the Fuzzy controller performed much better than the PID controller. The Fuzzy controller had a steady state absolute error of less than 1 mph or 4% while the PID controller was more than 4 mph or 16%. The overshoot for the Fuzzy controller was 2.5% compared to the PID controller's 16%. Since the Fuzzy controller performed better in Car 1 it was not surprising that it outperformed the PID controller in Car 2 as well. The fuzzy controller error stayed within 2 mph or 8% while the PID controller had a steady state error greater than 5 mph or 20%. The overshoot for the Fuzzy controller in Car 2 was also 2.4% while the PID controller in Car 2 was around 25%.

From the above results that the Fuzzy controller adapted better from Car 1 to Car 2 than the PID controller. This could be also predicted from past experience. The PID controller is fined tuned for a specific plant. Changing the plant will diminish

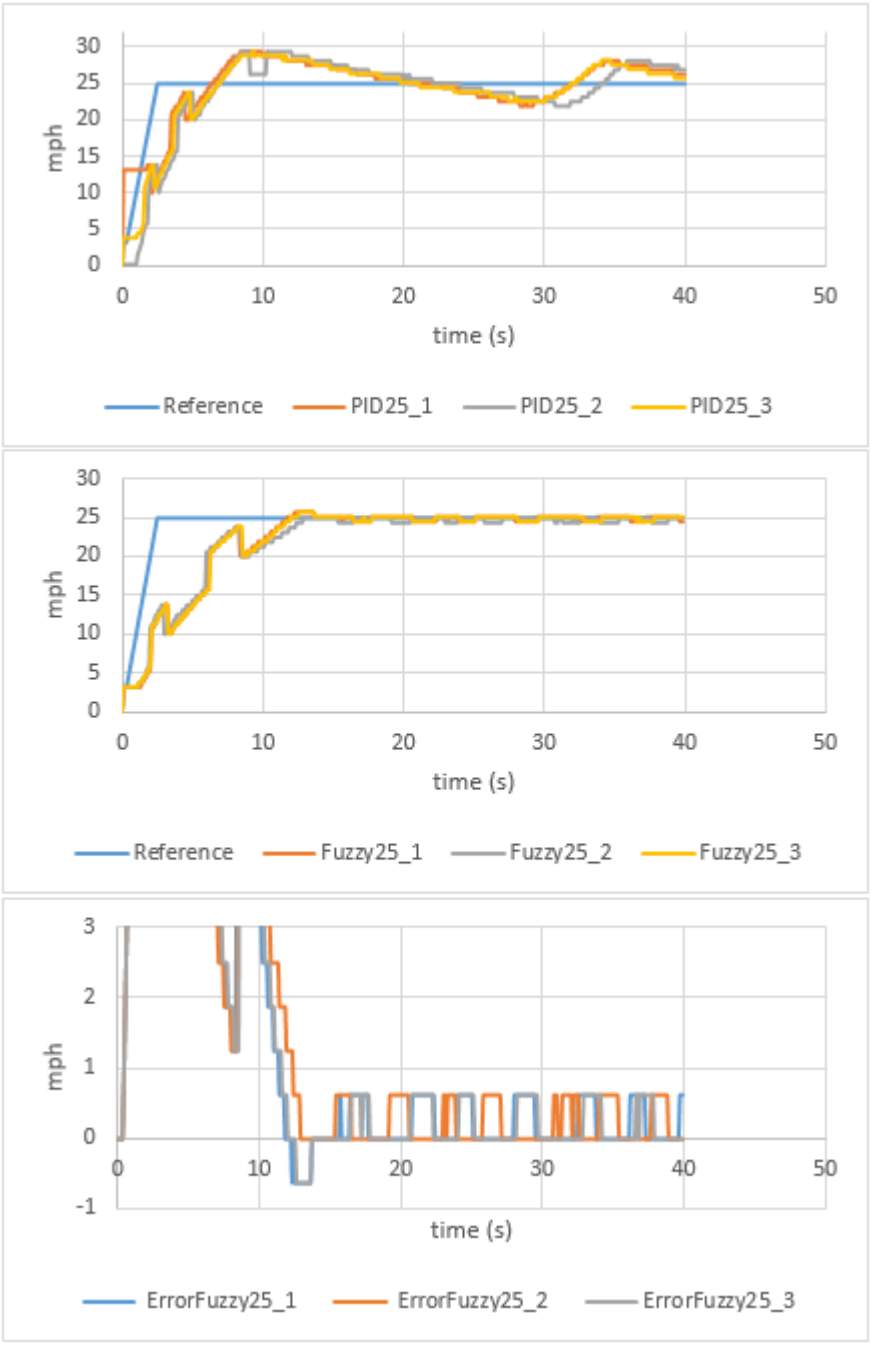


Fig. 5.9. PID vs. Fuzzy: Car 1 at 25 MPH

the performance. While the fuzzy controller's performance diminished as well, but not to the same extent. It can be reasoned that the linguistic approach makes the fuzzy controller easier to adapt from one plant to another.

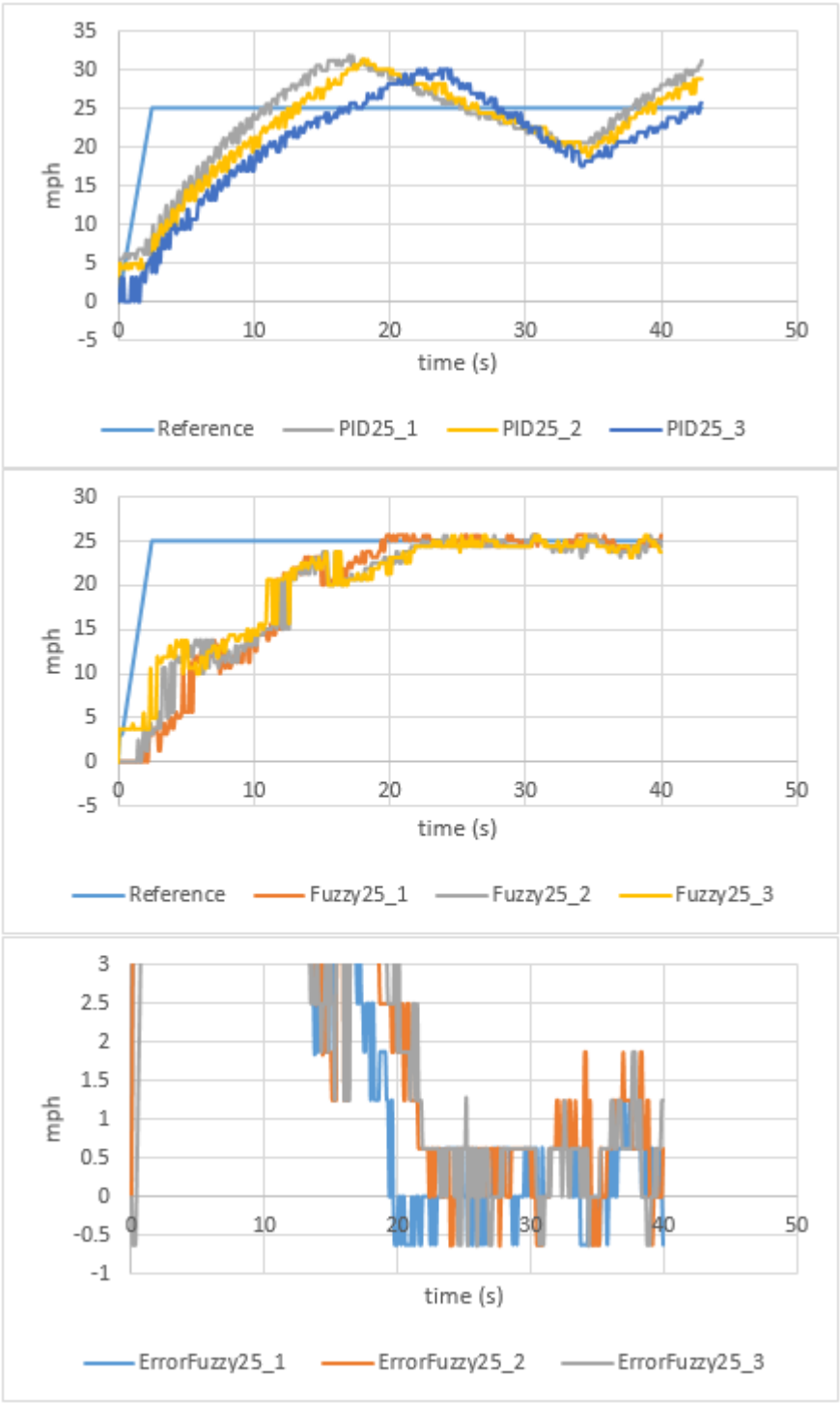


Fig. 5.10. PID vs. Fuzzy: Car 2 at 25 MPH

# 6. SUMMARY

## 6.1 Conclusion

In conclusion the fuzzy controller worked much better in both cars when compared to the PID controller. The fuzzy controller had a steady state error of 1 mph in Car 1 and 2 mph in Car 2, while the PID controller had more than 4 in both cars. The poor performance of the PID controller was surprising considering that the PID method is the most common in the automobile industry. Perhaps a more experienced controls engineer could have tuned the PID controller to achieve better performance. A more advanced PID algorithm might also improve the performance. The parallel linkage under performed on controlling over-shoot, especially with the PID controller. This in part due to the small distance between the accelerator and the brake pedal. This resulted in limited force being applied to the brake.

## 6.2 Future Work

There are many avenues for future work from this paper. One is improving the braking action design. One could improve either the parallel linkage design or add another actuator to the system that is dedicated to braking.

The implemented parallel linkage design did not work as well as expected. The main reason was because of the short distance between the accelerator and the brake pedal. This translated to a short travel distance of the brake plunger. It is possible that a more advanced parallel linkage design could be used to introduce more travel distance for the brake plunger. This would allow the designer to set the brake plunger in such a position so that it has a smooth braking action that would decrease the overshoot.

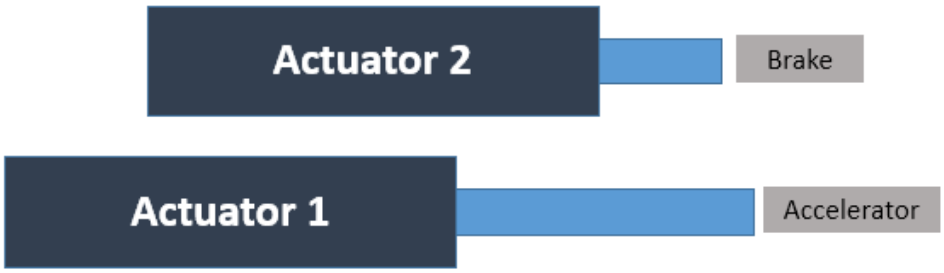


Fig. 6.1. Two Actuator System

Another option is to introduce another actuator to the system to control the braking action as depicted in Fig. 6.1. This system would be more expensive and more complex but has an upside of having very good control. The second actuator replaces the need for the parallel linkage mechanism. Both actuators would have their own control law. However, there would need to be some logic put in places so avoid the event where both the brake and the accelerator are being pressed at the same time. In other words, one actuator cannot extend forward until the other actuator is retracted and vise versa.

PID self-tuning is an area that would improve the implementation of the PID controller. As stated in the results section, the PID controller did not do as well expected and this is at least in part from controller not have an optimal tuning. A self-tuning algorithm could help with finding the optimal control parameters resulting in improved performance. This algorithm could also help with the car adaptability issue. It be used to find the optimal parameters any time the controller is moved from to another car.

Another potential improvement is adding an adaptive algorithm to the fuzzy controller. This could decrease the steady state even more, as well as improve its ability to adapt from one car to another.

Finally, another area that needs work is improving the Arduino fuzzy library. When comparing the output of MATLAB's Fuzzy controller and Arduino's Embedded

Table 6.1.
MATLAB and Arduino Fuzzy Control Output Discrepancy

| Input | | MATLAB Output | Arduino Output | Difference |
|---|---|---|---|---|
| Error | ΔError | | | |
| -1 | -1 | -0.823 | -0.817 | 0.006 |
| -0.9 | -0.9 | -0.817 | -0.812 | 0.005 |
| -0.8 | -0.8 | -0.804 | -0.799 | 0.005 |
| -0.7 | -0.7 | -0.796 | -0.791 | 0.005 |
| -0.6 | -0.6 | -0.811 | -0.806 | 0.005 |
| -0.5 | -0.5 | -0.821 | -0.815 | 0.006 |
| -0.4 | -0.4 | -0.671 | **-0.544** | 0.127 |
| -0.3 | -0.3 | -0.492 | **-0.635** | 0.143 |
| -0.2 | -0.2 | -0.217 | -0.217 | 0 |
| -0.1 | -0.1 | -0.119 | -0.131 | 0.012 |
| 0 | 0 | 0 | 0 | 0 |
| 0.1 | 0.1 | 0.119 | 0.108 | 0.011 |
| 0.2 | 0.2 | 0.217 | 0.217 | 0 |
| 0.3 | 0.3 | 0.492 | **0.608** | 0.116 |
| 0.4 | 0.4 | 0.671 | **0.424** | 0.247 |
| 0.5 | 0.5 | 0.821 | 0.815 | 0.006 |
| 0.6 | 0.6 | 0.811 | 0.806 | 0.005 |
| 0.7 | 0.7 | 0.796 | 0.791 | 0.005 |
| 0.8 | 0.8 | 0.804 | 0.799 | 0.005 |
| 0.9 | 0.9 | 0.817 | 0.812 | 0.005 |
| 1 | 1 | 0.823 | 0.817 | 0.006 |

Fuzzy Logic Library (eFLL) there was some discrepancies. These discrepancies are highlighted in Table. 6.1. The values are within a couple hundredths for most inputs but between -.45 and -.25 and between .25 and .45 the two outputs differ. These values were as much as two tenths off which in the case is significant. Future work would include finding the reason for this difference and fixing the issue and retesting the two cars to see if there is an improvement in the controller's performance. During the testing the actuator occasionally oscillated from the max position to min position and this fuzzy library error could be causing some of it.

# REFERENCES

# REFERENCES


[1] *Safe Driving with Cruise Control! - GM Fleet.* Warren, MI: General Motors, 2012, http://www.gmfleet.com/content/dam/gmfleet/global/master/nscwebsite/en/Home/Shared_Resources/PDFs/gmc1-12-03142-259-cruise-control.pdf, Last date accessed: May, 17, 2017.

[2] *Adaptive Cruise Control and Collision Warning - Ford Media.* Dearborn, MI: Ford Motor Company, 2012, https://web.archive.org/web/20131014210324/http://corporate.ford.com/doc/Adaptive_Cruise.pdf, Last date accessed: May, 17, 2017.

[3] U. Kiencke and L. Nielsen, *Automotive Control Systems.* Berlin, Germany: Springer-Verlag, 2005.

[4] M. L. Boas, *Mathematical Methods in the Physical Sciences*, 3rd ed. Indianapolis, IN: Wiley Publishing Inc, 2005.

[5] K. J. Åström, *Control System Design.* Santa Barbara California: Karl Johan Åström, 2002.

[6] G. K. Batchelor, *An Introduction to Fluid Dynamics.* Cambridge, UK: Cambridge University Press, 2000.

[7] J. M. Parr and C. L. Phillips, *Feedback Control Systems*, 5th ed. Upper Saddle River New Jersey: Pearson Education, Inc, 2011.

[8] E. D. Ruiz-Rojas, J. L. Vazquez-Gonzalez, R. Alejos-Palomares, A. Z. Escudero-Uribe, and J. R. Mendoza-Vzquez, *Mathematical Model of a Linear Electric Actuator with Prosthesis Applications.* New York New York: IEEE, 2008.

[9] K. J. Åström and T. Hägglund, *PID Controllers: Theory, Design, and Tunning*, 2nd ed. Research Triangle Park, NC: Instrument Society of America, 1995.

[10] K. M. Passino and S. Yurkovich, *Fuzzy Control.* Chicago, IL: Addison-Wesley Longman, Inc., 1998.

[11] K. Mahmud and L. Tao, *Vehicle Speed Control Through Fuzzy Logic.* Xian 710072, P.R. China: 2013 IEEE Global High Tech Congress on Electronics, 2013.

[12] D. L. Jenkins and K. M. Passino, *Introduction to Nonlinear Analysis of Fuzzy Control Systems.* Clifton, VA: IOS Press, 1999.

[13] M. B. Trabia, L. Z. Shi, and N. E. Hodge, *"A Fuzzy Logic Controller for Autonomous Wheeled Vehicles", Mobile Robots, Moving Intelligence.* Rijeka, Croatia: InTech, 2006.

[14] K. Ang, G. Chong, and Y. Li, *PID Control System Analysis, Design, and Technology.* New York, NY: IEEE Trans Control Systems Tech, 2005.

# APPENDICES

# A. PARAMETERS

## A.1 Electric Actuator Motor Parameters

The following parameters are from Tolomatic's ICR SmartActuator (config: ICR20S BN05 SM10 LMI SV1P CPS CNC1 MET FFG)

| Parameter | value | Description |
|---|---|---|
| $J_m$ | $3.9545 \cdot 10^{-5}$ kg$\cdot m^2$ | Motor inertia |
| $k_b$ | .1198 V$\cdot$ s/rad | Fem Constant |
| $R_a$ | 0.59 $\Omega$ | Armature Resistance |
| $L_a$ | $5.2 \cdot 10^{-4}$ H | Armature Inductance |
| $k_\tau$ | .1198 N$\cdot$ m/A | Motor torque constant |
| $D_m$ | $1 * 10^{-7} N \cdot m \cdot s/rad$ | Air and bearing friction |
| $L$ | 5.08 mm/rev | Ball-screw lead step size |

## A.2 Drag Force Parameters

| Parameter | value | Description |
|---|---|---|
| $c_d$ | 0.4 | Drag Coefficient |
| $A$ | 3 $m^2$ | Car Front Cross-sectional Area |
| $\rho$ | 1.225 $\frac{kg}{m^3}$ | Air Density @ 15$C^\circ$ |

# B. ALL TESTING AND EXPERIMENTAL RESULTS

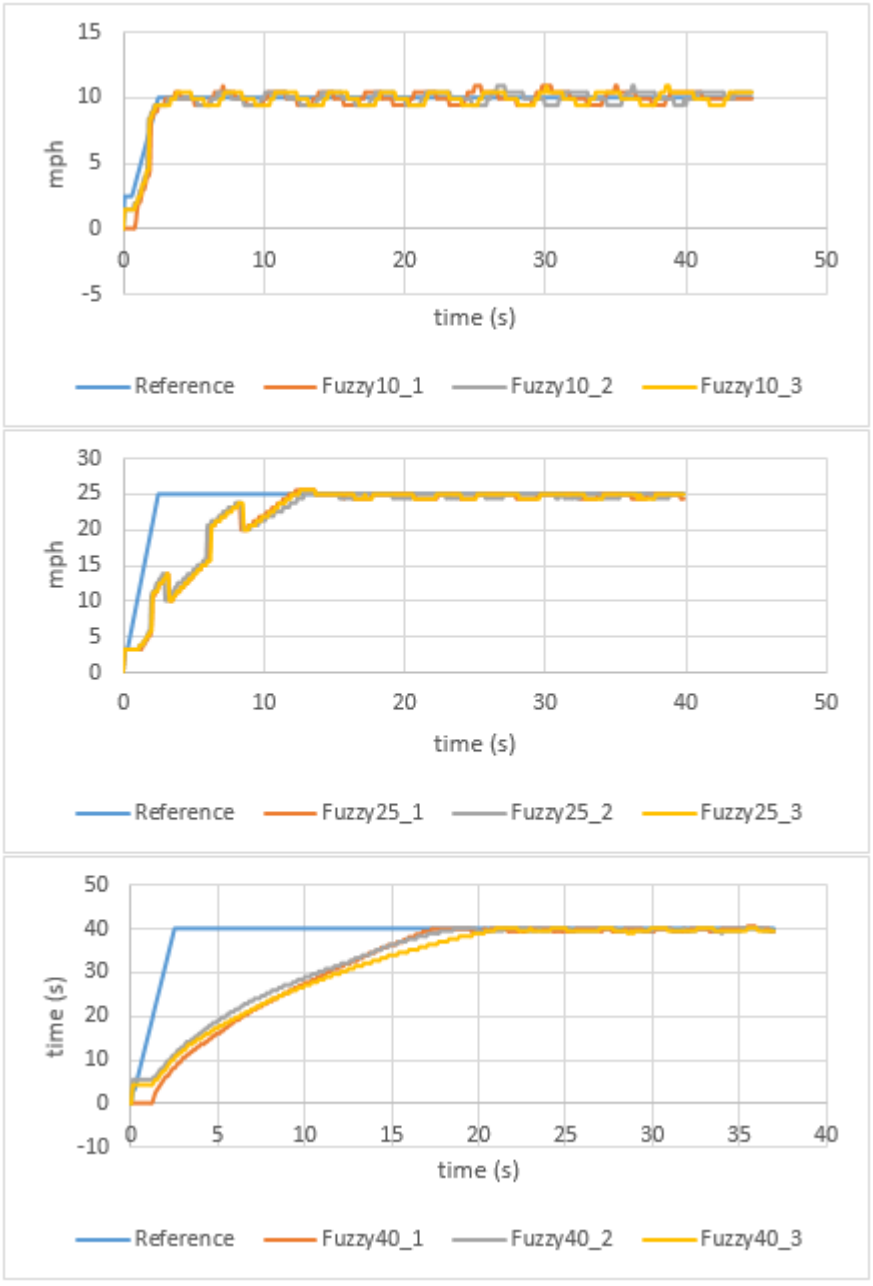


Fig. B.1. Car 1 Fuzzy Control Results

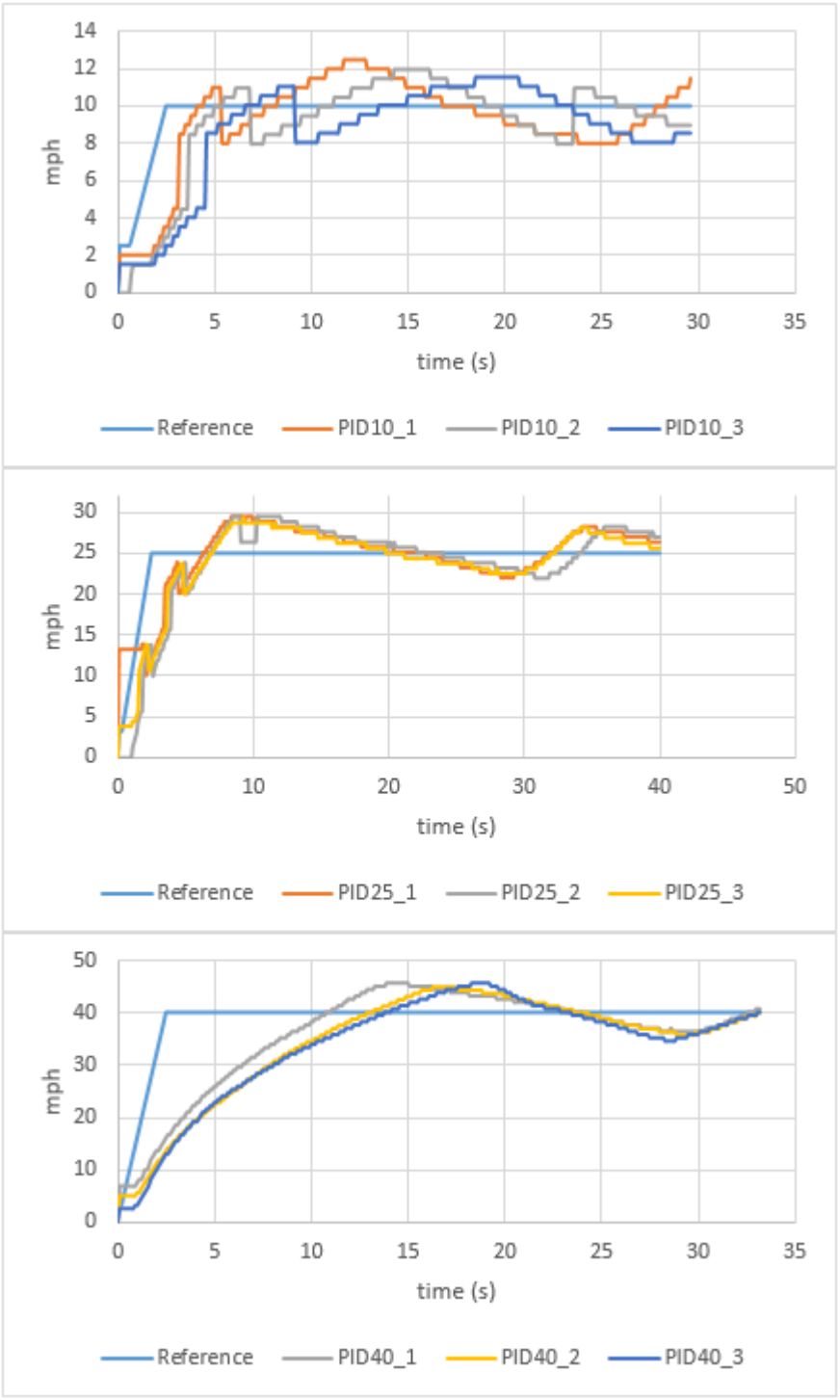


Fig. B.2. Car 1 PID Control Results

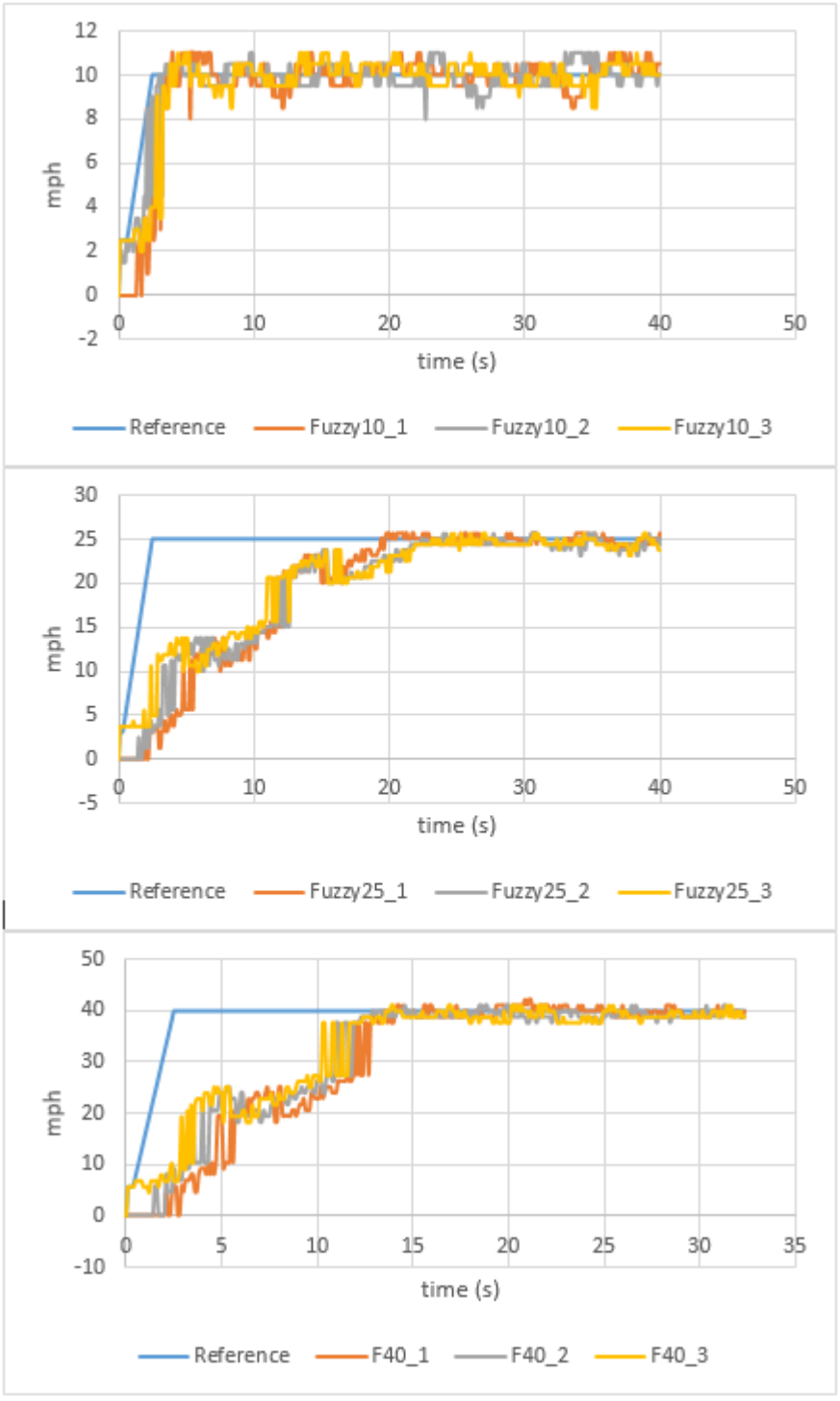


Fig. B.3. Car 2 Fuzzy Control Results

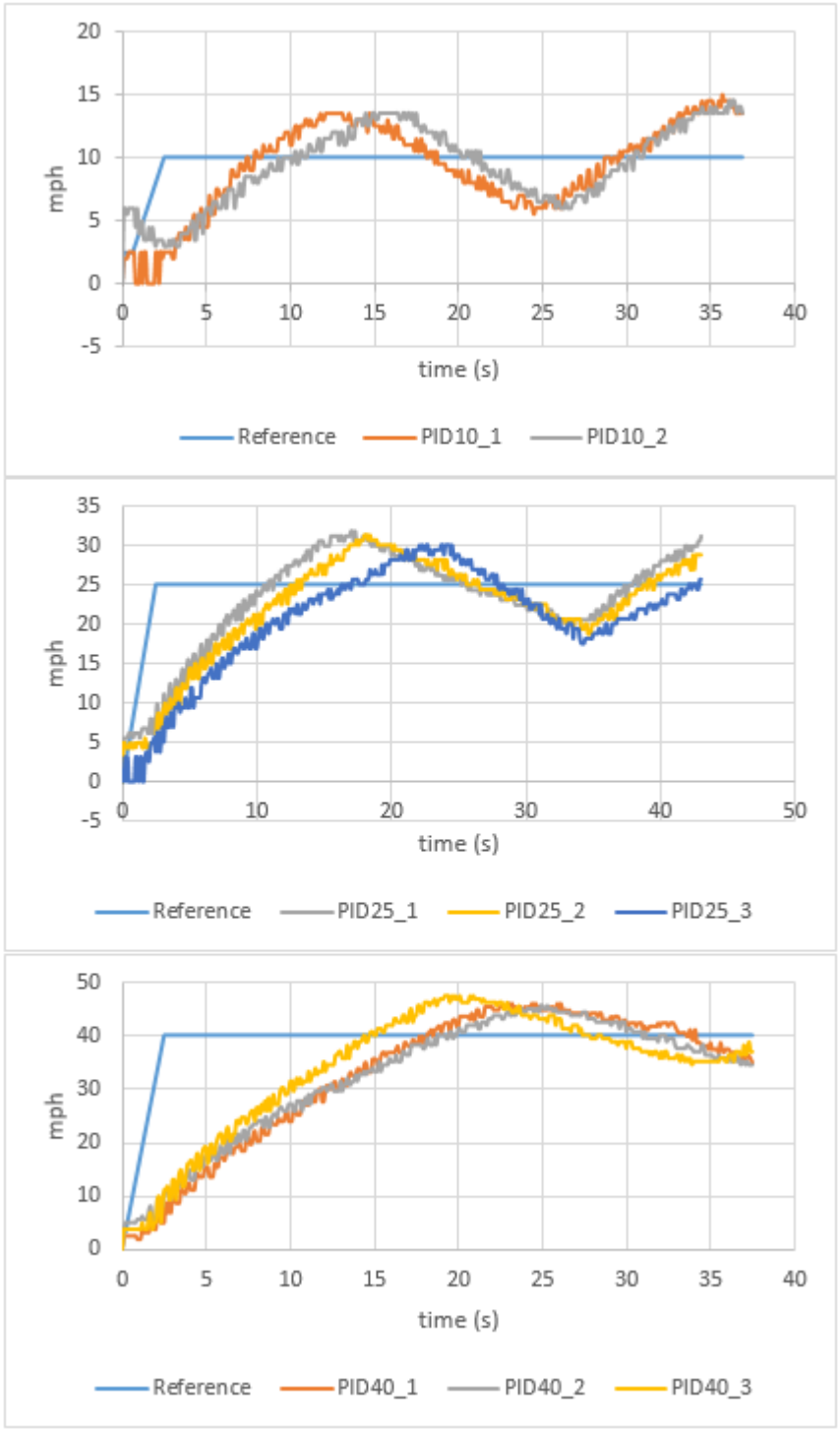


Fig. B.4. Car 2 PID Results